Rubik's Cube

Speedcubing and Principles

By Thomas Du

Copyright

Copyright © 2026 Thomas Du. All rights reserved.

No part of this book may be reproduced, distributed, or transmitted in any form or by any means without prior written permission from the author, except for brief quotations and other uses permitted by applicable law.

This book is an educational introduction to speedcubing and the mathematical principles behind the Rubik's Cube. Explanatory prose was written primarily by the author. Selected algorithms, case organizations, examples, diagrams, photographs, and other materials were adapted from or based on the sources identified in the **Source-Use Notes**, **References**, and **Image and Diagram Credits** sections.

Rubik's Cube and other puzzle, organization, product, and brand names are trademarks or property of their respective owners. Their use in this book is for identification and educational purposes only and does not imply sponsorship or endorsement.

Some cover, table-of-contents, and chapter-opening artwork was created using generative artificial-intelligence tools under the author's direction.

The author has made reasonable efforts to identify and credit external sources and to verify the accuracy of the material. Corrections or requests concerning attribution may be sent to: **duzhuoyuan@gmail.com**.

First edition, 10 July 2026.

Table of Contents

Preface i, ii, iii

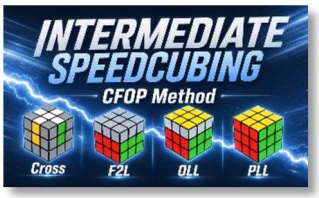
1 Getting Started 1

1.1 Notation and Convention 2
1.2 3×3×3 Beginner's Tutorial 4
1.3 2×2×2 Cube Introduction 10

Overview: CFOP Speedcubing Path 13

2 Intermediate Speedcubing 14

2.1 First Two Layers (1) 15
2.2 OLL and PLL Algorithms 21
2.3 CFCE & Roux 34

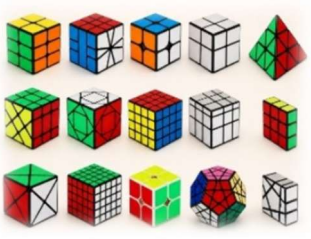


3 Advanced Speedcubing 44

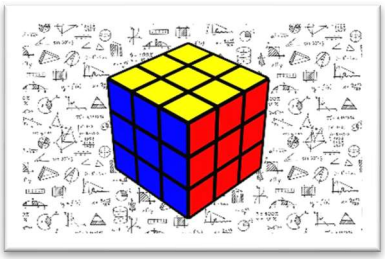
3.1 First Two Layers (2) 45
3.2 Last Slot Techniques 57
3.3 Advanced Methods 65

4 Other Events 79

4.1 Big Cubes 80
4×4×4, 5×5×5, 6×6×6, 7×7×7
4.2 Twisty Puzzles 89
Pyraminx, Skewb, SQ1, FTO, Megaminx



5 Cube Theory 103



Introduction: *Why Cube Theory Matters*

5.1 Rubik's Cube as a Group 104

5.2 Legal States and the Invariants 106

5.3 Group Concepts and Domino Reduction 108

5.4 Cayley Graph and God's Number 111

5.5 Commutator and Conjugation 114

5.6 3BLD Introduction: M2/OP 118

5.7 3FMC Introduction: NISS 123

Chapter Summary 126

Source-Use Notes 128

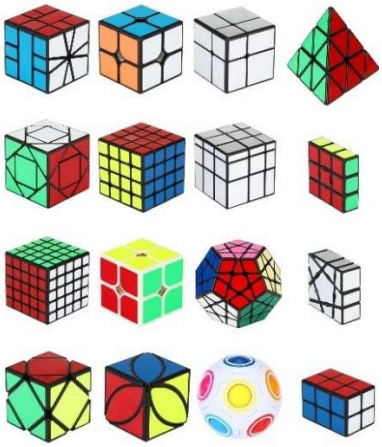
References 129

Image and Diagram Credits 131

Preface

Introduction

The Rubik's Cube is a popular and interesting 3D combination puzzle invented by Ernő Rubik in 1974. Although it began as the original 3×3×3 cube, it has since evolved into many different sizes and shapes, including the 2×2×2, 4×4×4, 5×5×5, Pyraminx, Megaminx, FTO, Skewb, and Square-1. While these puzzles may look simple in appearance, solving them requires pattern recognition, memory, spatial awareness, logic, and problem-solving skill.



The Rubik's Cube has also become a global competitive puzzle. Over the years, many solving methods have been developed, including the beginner's layer-by-layer method, CFOP, Roux, and ZZ. These methods focus on improving efficiency and reducing the number of moves needed to solve the cube. One famous idea in cube theory is "God's Number," the maximum number of moves needed to solve any scrambled 3×3×3 cube, which is 20.

This book is a comprehensive guide to Rubik's Cube solving, with an emphasis on the 3×3×3 CFOP Method and the mathematical principles of the cube. This book aims not only to help readers solve the Rubik's Cube, but also to help them improve their efficiency, explore the wider world of twisty puzzles, and understand the principles beneath the surface.

The content in this book is presented mainly through written instructions, algorithms, and supporting pictures. The instructions explain the purpose of each step, the algorithms provide exact move sequences, and the pictures help readers recognize cube cases and follow the solving process visually. Some content in chapter 5 is presented by mathematical expressions.

Most of the algorithms and examples are designed for right-handed solvers, since this is the most common convention in cubing tutorials, but a small number of left-handed algorithms are also included where they are useful.

The World Cube Association

The World Cube Association, often called the WCA, governs official speedcubing competitions around the world. Its events include the standard 3×3×3 cube, larger N×N×N cubes, one-handed solving, blindfolded solving, Fewest Moves, Clock, Megaminx, Pyraminx, Skewb, and Square-1. Face-Turning Octahedron, or FTO, has been announced as a new official event and will become available at WCA competitions beginning in 2027.

Competitive cubing has grown greatly since the first official Rubik's Cube World Championship in 1982, where the winning time was 22.95 seconds. Today, the 3×3×3 world record is under 3 seconds, showing major improvements in solving methods, turning speed, and finger tricks. Famous speedcubers such as Max Park, Feliks Zemdegs, Yusheng Du, Tymon Kolasiński, Yiheng Wang, Xuanyi Geng, and Luke Garrett have helped push the limits of human solving speed.

WCA competitions are not only about speed. Some events test memory, some focus on efficiency, and others involve different puzzle structures. Together, they show that cubing is a worldwide activity built around logic, practice, competition, creativity, and perseverance. Readers should check *worldcubeassociation.org* for the most current information.

CubeRoot and Other Sources

This book makes extensive use of CubeRoot, created by Ruimin Yan, as a reference for speedcubing methods, algorithm collections, case organization, and diagrams. Some materials were also adapted from tutorials and short-form videos published by Ruimin Yan through other online platforms, including WeChat.

Most explanatory prose in this book was written by the author, while selected algorithms, diagrams, examples, and method structures were adapted from the sources identified in the **Source-Use Notes**, **References**, and **Image and Diagram Credits** sections.

I sincerely thank Ruimin Yan for creating and sharing these valuable resources with the cubing community.

Content

The first chapter, **Getting Started**, begins with notation and cube conventions, the basic language for cube algorithms. This chapter then introduces the beginner's method for solving the 3×3×3 cube, followed by a short introduction to the 2×2×2 cube. The beginner's tutorial includes useful improving techniques and it might be helpful even for readers who have certain solving skills.

The second chapter, **Intermediate Speedcubing**, focuses on faster 3×3×3 solving methods. It covers the intermediate CFOP system, including First Two Layers, OLL and PLL algorithms. It also introduces the CFCE Method and the Roux Method as other options. This chapter is intended to provide readers with a comprehensive guide in 3×3×3 speedcubing.

The third chapter, **Advanced Speedcubing**, provides expert-level techniques in all solving stages of the CFOP Method: Multi-angle F2L, Non-standard F2L, Lookahead, OLS, VH Method, and 2GLL. Additionally, it introduces some other advanced solving structures such as ZZ and Petrus.

The fourth chapter, **Other Events**, introduces larger cubes such as 4×4×4, 5×5×5, 6×6×6, and 7×7×7, as well as twisty puzzles such as Megaminx, Pyraminx, Skewb, Square-1, and FTO. These tutorials give readers a broad view of the main puzzle types used in speedcubing.

The fifth chapter, **Cube Theory**, looks deeply at the cube from the mathematical perspective. It discusses topics such as group theory, legal states, invariants, Lagrange's Theorem, Domino Reduction, Cayley graph, God's Number, commutator, conjugation. It also introduces the fundamental methods for 3BLD and 3FMC cubing events to demonstrate that cubing is not only about speed, but also about structure, logic, memory, and efficient problem solving.

Recommendation for different skill levels

Readers who are new to cubing should start from the first chapter, including the notation, convention, and beginner's tutorial. As an extension, they may learn some other easy cubes such as 2×2×2, Pyraminx, and Skewb.

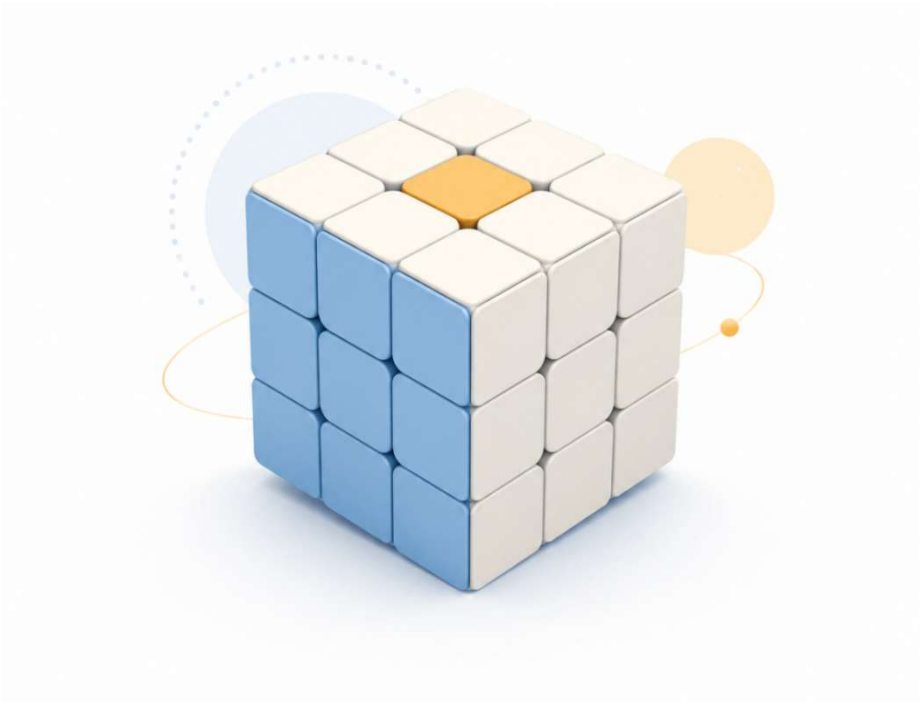
Readers who already know how to solve the 3×3×3 cube can use the beginner's tutorial to improve their solving skills, and explore some easy topics in this book that they are interested in.

Readers who are proficient at the 3×3×3 cube (15-30 seconds) are recommended to study the second chapter to build a comprehensive CFOP solving system. At the same time, they are able to access some other topics in this book, such as cube theory, big cubes, and harder twisty puzzles.

Expert readers who can solve the 3×3×3 cube in under 15 seconds may use this book as a reference and focus on topics that match their proficiency, goals, and interests. Advanced topics in chapter 3 and chapter 5 are only recommended for these readers.

The prerequisites for each chapter are indicated on the title pages.

1 Getting Started



This chapter is the starting point of your cubing journey. You will first learn the notation and conventions used to describe cube moves, then use them to solve the 3×3×3 cube with a beginner-friendly method. The chapter also introduces the 2×2×2 cube, showing how simple ideas from the 3×3×3 cube can lead to solving new puzzles.

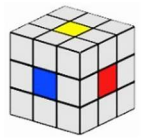
Prerequisites: None

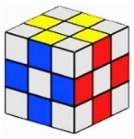
1.1 Notation and Convention

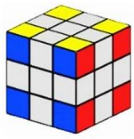
This book follows the standard notation system for the 3×3×3 cube. The notations for other cube types are specified in chapter 4.

Structure & Color

The 3×3×3 cube consists of 6 center pieces, 12 edge pieces, and 8 corner pieces. The relative positions of 6 center pieces are fixed. A key point is that white and yellow, blue and green, red and orange are **opposite colors**. This book uses standard color scheme, as shown below.


Center pieces


Edge pieces


Corner pieces

Notation

A notation system is used to describe the turns of the cube.

Face Turns

Each move means to turn that side clockwise, as if you were facing that side. Adding an apostrophe indicates counterclockwise. For example, U' (U prime) turns the opposite direction as U. Adding a "2" indicates turning twice.

U (Up)

D (Down)

R (Right)

L (Left)

F (Front)

B (Back)

U'

D'

R'

L'

F'

B'

U2

D2

R2

L2

F2

B2

Wide Turns

u

d

r

l

f

b

Slice Turns

E, M, S follow D, L, F directions by convention.

E (Equator)

M (Middle)

S (Standing)

Rotations

x, y, z follow R, U, F directions by convention.

x

y

z

Other Conventions

Scramble Orientation

In all 3×3×3 scramble formulas, it is assumed that white is facing up and green is facing front, as the picture on the right shows.

Algorithms

An algorithm is a sequence of turns. Example: A = R U R' U'

Repetition: We use abbreviations for repeating algorithms. Example: (X Y)2 = X Y X Y

Grouping: We use parentheses to group turns. Grouping is beneficial for memorizing, and is sometimes used to indicate a change of grip. Example: (X Y) (Z W) = X Y Z W

Direction: R2/ R2' and M2/M2' refer to the same move but different hand movements. R3 occurs several times and it refers to an actual R3 move for grip consistency.

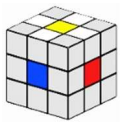
1.2 3×3×3 Beginner's Tutorial

The best method for 3×3×3 beginners is the **LBL Method** (Layer-by-Layer) since ideas are easy to learn and the algorithms are simple. By practicing, you can become more proficient and be able to reach a solving time within one minute. Then, you may start learning the improving skills. The improved method condenses the F2L process into 3 steps, and helps achieve 4-look-last-layer (4LLL), meaning that you only need to perform four algorithms to solve the last layer (the basic method involves repeating algorithms several times).

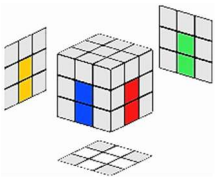
Steps

First Two Layers (F2L)

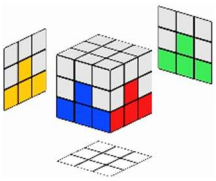
White Flower



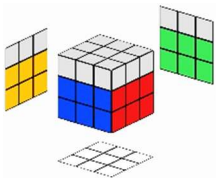
White Cross



First Layer

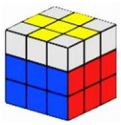


Second Layer

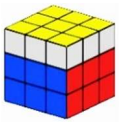


Last Layer (LL)

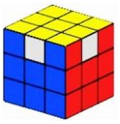
Edge Orientation



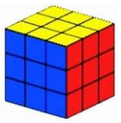
Corner Orientation



Corner Permutation



Edge Permutation



Basic LBL Method

White Flower

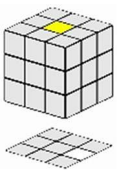
White is typically used as the bottom layer because it is easier to track. Turn all 4 white edge pieces to the U-layer and make sure the white sides face up. Examples:

R

L'

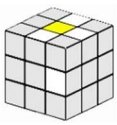
F R

F' R



R2

U' R or U R

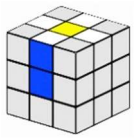


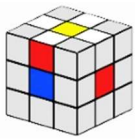
U2 R

F U' R

White Cross

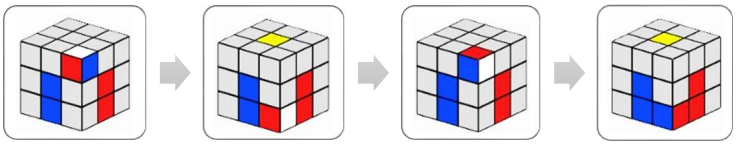
Rotate the U layer to align the edge piece with the center, then turn it 180° to complete one edge. Repeat this process until all the white edges are aligned. Examples:

 (Paired) **F2**

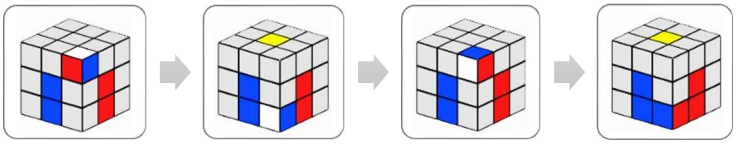
 (Not paired) **U' R2**

First Layer

Complete the 4 white corners. For any case in the process chart, keep performing the algorithm until the corner is solved. If there are no corners in the U layer, apply one of the algorithms to bring it up.



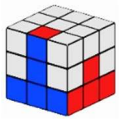
Alg 1 **R U R' U'**



Alg 2 **U R U' R'**

Second Layer

Complete the 4 edges without white or yellow. If there is no edge in the U layer, apply one of the algorithms to bring the target edge up to the U layer.



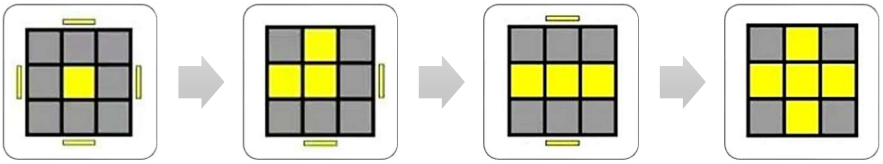
U R U' R' y U' L' U L (Alg 3)

y U' L' U L y' U R U' R' (Alg 4)

(Alg 3)

Note: The pictures below show only the top layer of the cube. Rotate the U layer freely to ensure the structure is in the right orientation. The arrows indicate performing the algorithm and transforming one structure to another, until it is solved.

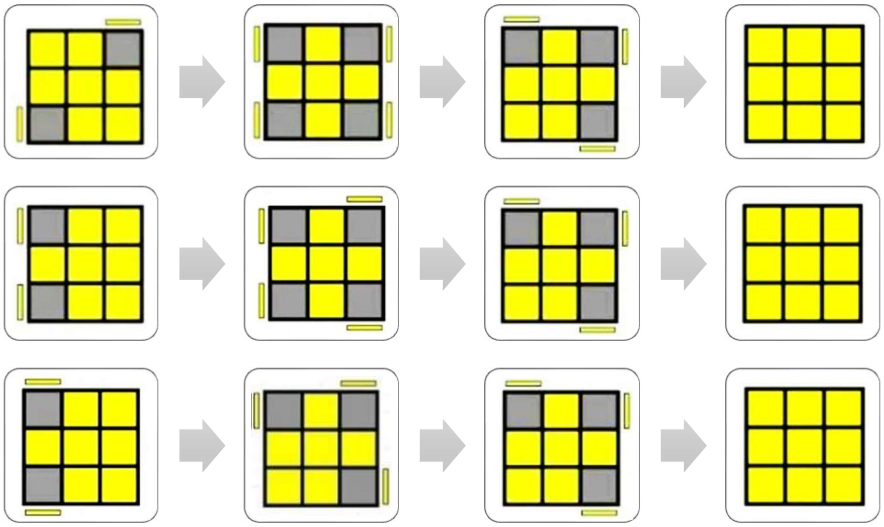
Edge Orientation of Last Layer (EO)



Alg 5 **F R U R' U' F'**

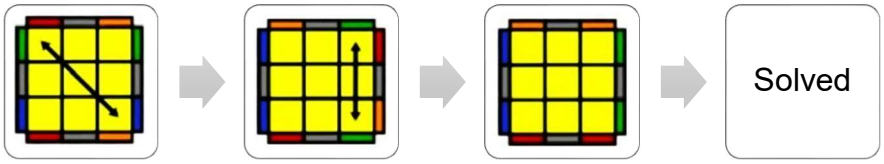
Corner Orientation of Last Layer (CO)

There are 7 possible cases. Pay attention to the orientation. **Alg 6** **R U R' U R U2 R'**



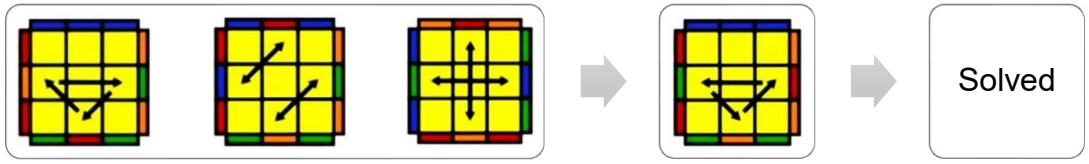
Corner Permutation of Last Layer (CP)

Alg 7 **(R U R' U') (R' F R2 U' R' U') (R U R' F')**



Edge Permutation of Last Layer (EP)

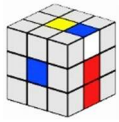
Alg 8 **M2' U M U2 M' U M2'**



Improving Skills

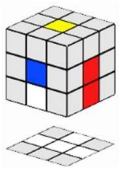
Cross Skills

Instead of making the white flower first, you can immediately place the edges in their correct positions. It is a good habit to keep the white face down. Another skill is that you don't need to keep all edges aligned with the centers, but keep their sequences: red, blue, orange, green (clockwise). The picture on the right is a great example. You can also try to solve more than one edge at one time. These skills are discussed in the F2L (1) lesson in the second chapter.

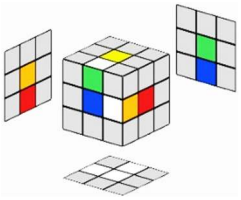


R' F R

I U' I'



F' D R' D'



R' F2 D2

F2L (First Two Layers) Skills

F2L is considered the most important phase of the solution due to its complexity. These two skills will help you get a smoother transition between the beginner's method and the CFOP system.

Free Pair

The intermediate method of F2L is to solve the edge and corner pair in 1 algorithm. The 4 cases below are called "Free pair", which we can solve the corner and the edge together easily.

Connected Form

U R U' R'

F R' F' R or y U' L' U L

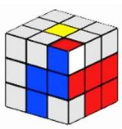
Separated Form

R U R'

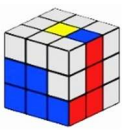
F' U' F

Keyhole / 8355

When the corners or the edge of a slot is solved while the other isn't, we can use an empty slot (usually by D moves) to solve it as if there is no solved piece. In the second case, D' moves the solved corner away so we can solve the edge easily. The **8355 Method** utilizes keyholes: Solving 3 edges first, use the last slot as the keyhole to solve the 4 corners, and finally solve the last edge. This method uses D moves for solving corners instead of cube rotations.



D' L' U L' D



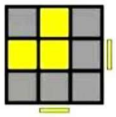
D' U R U' R'

U R U' R' D' R U' R' D

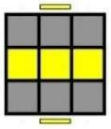
4LLL Algorithms (4-Look-Last-Layer)

Memorizing several additional formulas helps you solve the last layer much more efficiently.

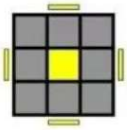
Edge Orientation (EO)



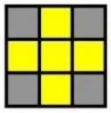
F R U R' U' F'
Probability = 1/2



F U R U' R' F'
Probability = 1/4

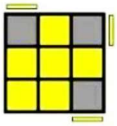


F (R U R' U') S (R U R' U') f'
Probability = 1/8

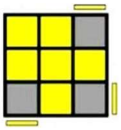


Solved
Probability = 1/8

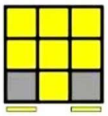
Corner Orientation (CO)



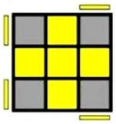
R U R' U R U2 R'
Probability = 4/27



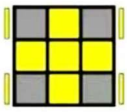
R' U' R U' R' U2 R
Probability = 4/27



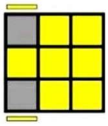
R2 D R' U2 R D' R' U2 R'
Probability = 4/27



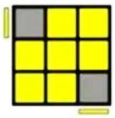
R U2' R2' U' R2 U' R2' U2' R
Probability = 4/27



R' U' R U' R' U R U' R' U2 R
Probability = 2/27



r U R' U' r' F R F'
Probability = 4/27



F' r U R' U' r' F R
Probability = 4/27

Solved
Probability = 1/27

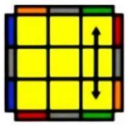
*Three properties of the Rubik's Cube

You might notice that there is no case where only 1 or 3 edges are oriented, and there is no case where only 1 corner is twisted. This is one of the special properties of a 3×3×3 cube:

1. An edge piece cannot flip itself.
2. A corner piece cannot twist itself in either direction.
3. A pair of pieces cannot swap without any movements of other pieces.

These properties will be discussed in the Cube Theory chapter.

Corner Permutation (CP)



(R U R' U') (R' F R2 U' R' U') (R U R' F')
Probability = 2/3

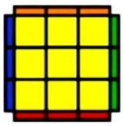
F (R U' R' U') (R U R' F') (R U R' U') (R' F R F')
Probability = 1/6

Solved
Probability = 1/6

Edge Permutation (EP)

M2' U M U2 M' U M2'
Probability = 1/3

M2' U M2' U2 M2' U M2'
Probability = 1/12



Solved
Probability = 1/12

M2' U' M U2 M' U' M2'
Probability = 1/3

M2' U M2' U M' U2 M2' M'
Probability = 1/6

***How to calculate the probability of each case in EO, CO, CP, and EP?**

The probability of each case depends on the number of different configurations it can have. When you rotate the U layer by 90°, 180°, or 270°, the positions of the pieces change, leading to different possible forms.

For example, in the EO (Edge Orientation) part, when we rotate the U layer in the first case, there are 4 possible configurations. In the second case, rotating the U layer 180° results in the same position as before, reducing the number of configurations. Similarly, rotating the U layer by 90° or 270° produces the same positions, allowing only 2 possible configurations in the second case. In the third and fourth cases, regardless of how you rotate the U layer, the configuration remains fixed, allowing for only 1 possible position.

Considering these factors, we can calculate that there is a total of 8 possible configurations. From this, we can determine the probability of the first case as 4/8, or 1/2.

Fun facts: the probability of an OLL (orientation of last layer) skip is 1/216, the probability of a PLL (permutation of last layer) skip is 1/72, the probability of a last layer skip is 1/15552.

1.3 2×2×2 Cube Introduction

A 2×2×2 cube can be solved using just R, U, and F moves. For smoother hand tricks, algorithms may include multiple generators or cube rotations. The LBL method for the 3×3×3 cube is still applicable; however, the **Ortega Method** is more efficient for beginners than the LBL method.

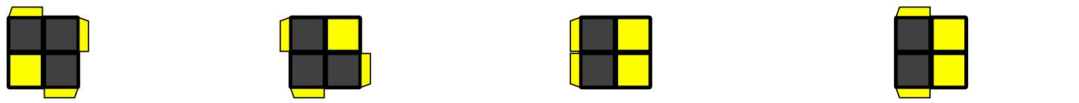
FF (First Face)

You can start with any color as your first face. During the inspection, plan how to complete ¾ of the face and predict the position of the last yellow corner. Use the 5 algorithms below to solve it, and turn D to position the slot correctly. Also, predict the permutation of the D layer.

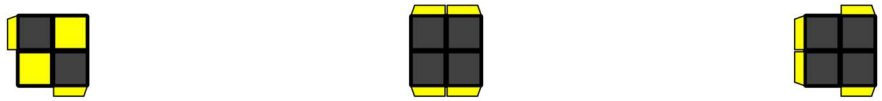
F' R' F R | **F R U' R'** | **R' U' R** | **R U R'** | **R2 U' R2'**

OLL (Orientation of Last Layer)

The positions are the same as on the 3×3×3 cube, but the algorithms are different.



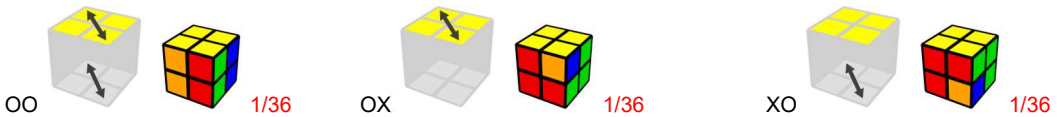
R U R' U R U2' R' | **R U2' R' U' R U' R'** | **F (R U R' U') F'** | **(R U R' U') (R' F R F')**



F' (R U R' U') R' F R | **R2' U2' R U2 R2'** | **R U' R2' U R2 U R2' U' R**

PBL (Permutation of Both Layers)

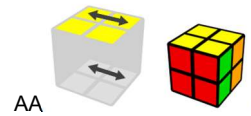
Turn the layers into the correct position, then apply the algorithms. Probabilities are indicated.



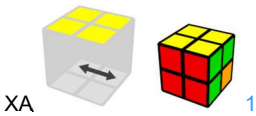
R2' F2 R2

(R U' R' U') F2 U'
(R U R' U) F2

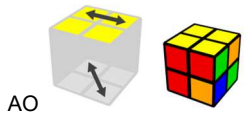
(R U' R' U') F2 U'
(R U R' U) R2 B2



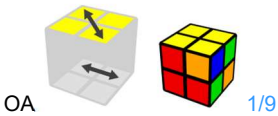
R2 U' B2 U2' R2' U' R2
y2 R2 U' R2' U2' F2 U' R2
y' F2 U' F2 U2' R2 U' R2'
y F2 U' R2 U2' B2 U' R2'



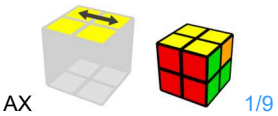
(R U' R' U') F
(R2 U' R' U) R' F' R
y (R2 U' R' U) (R' F2 R F') R' F2
y2 (R2 B2 U R2' U') (R2 U R2' U' R2)



R' F (R' F2 R U' R)
y2 (R' U R' F2) R F' R



R F' R B2 R' F R'
y (R2 U R2' U')2 R2
y2 (R U' R' U') R' F2 (U' R U R)



U (R' F R F') (R U2' R' U) R U2' R'

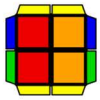
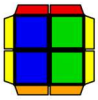
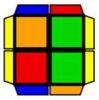
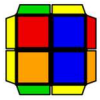
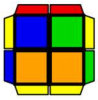
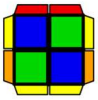
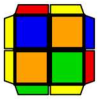
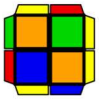
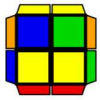
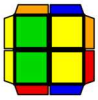
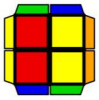
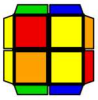
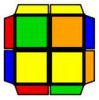
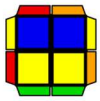
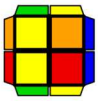
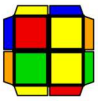
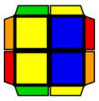
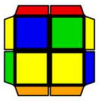
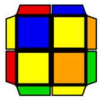
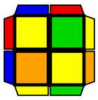
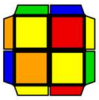
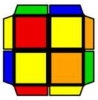
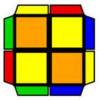
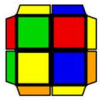
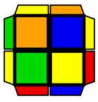
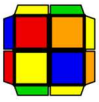
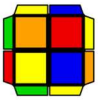
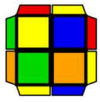
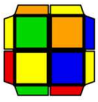
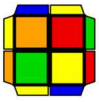
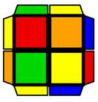
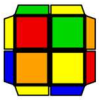
The advanced LBL Method

The advanced layer-by-layer includes only 2 steps: bottom layer and top layer. It can be much faster if you can both plan the first layer efficiently and memorize all the CLL algorithms. This is the starting point to become a 2×2×2 cube expert.

The CLL algorithms are on the next page.

Note: H, π, U, T, L, S+, S- are the 7 cases of CLL; X means the corners are permuted; D means the diagonal corners need to be swapped; R, L, F, B are all adjacent swaps.

CLL Algorithms (40)

	X	D	R	L	F	B
H	 R2 U2 R U2' R2'	 R U' R' (F R' F' R2 U' R') (F R' F' R)	 (R U R' U R U R') (F R' F' R)		F (R2 U' R2' U') (R2 U R2') F'	
π	 R U' R2' U R2 U R2' U' R	 R' U' (R' F R F') (R U' R' U2 R)	(R' F R F') (R U' R' U') R U' R'	 (R U' R U') (R' U R' F) R2 F'	 (R U2' R' U') (R U R' U2') (R' F R F')	 (L' U2 L U) (L' U' L U2) (L F' L' F)
U	 (R' U' R2 U) (R' U2 R U2') R' U R'	 F (R U R' U') F'	 (R U2 R U' R' F) R' F2 U' F	 (F R' F' R) U' (R U' R' U2) R U' R'	 (R U' R2' F R F') (R U R' U') R U R'	(L' U L2 F' L' F) (L' U' L U) L' U' L
T	 (R' U R U2') (R2' F' R U') R' F2 R2	 (R U2' R2' F R F') (R U' R' U) R U2' R'	(L' U' L U) (L F' L' F)	 (R U R' U') (R' F R F')	 (R U R' U2') (R U R' U) (R' F R F')	 (R' U R U2') (R2' F R F' R)
L	 R2' U' (R U2' R' U2) R U' R2'	 (R U2' R2' F R F') R U2' R'	 (F R' F' R) (U R U' R')	 F' (R U R' U') R' F R	(R U' R' U) (R U' R') (F R' F' R2 U R')	 R' F' (R U R' U') (R' F R2 U' R' U2 R)
S+	 R U R' U R U2' R'	 R' F R2 F' U' R' U' R2 U R'	 (F R' F' R) (U2 R U2' R')	 R U' R' (F R' F' R)	(R' F' R2 U R' F') (R' F R2 U' R')	(L' U2 L U2) (L F' L' F)
S-	 R' U' R U' R' U2 R	 L F' L2 F U L U L2 U' L	 (F' L F L') (U2' L' U2 L)	 (R' F R F') R U R'	(R U R2' F' R F) (R U' R2' F R)	 (R U2' R' U2) (R' F R F')

Overview: Speedcubing Path

The CFOP Method (Fridrich Method)

CFOP stands for **Cross – F2L – OLL – PLL**. It is an advanced layer-by-layer method designed for speed. The method was developed and refined by many speedcubers over time, and it became one of the dominant methods in competitive speed.

The CFOP method solves the cube fast because it divides the solve into logical stages. The cross creates a foundation, F2L solves the first two layers together, OLL orients the last layer, and PLL permutes the last layer pieces into their final positions. Compared with entry-level, CFOP reduces repeated steps, improves recognition, and leaves space for optimization.

The CFOP Method is efficient because it provides:

- Fewer moves, shorter algorithms
- A clear and predictable solving structure
- Patterns that are easy to recognize
- Good support for lookahead and advanced techniques

Learning Path

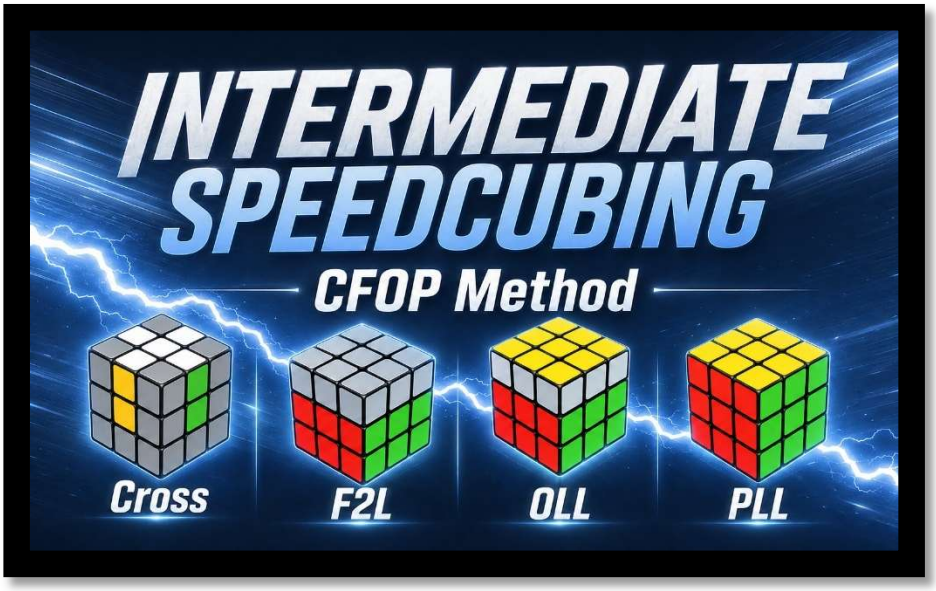
To start learning CFOP, it is strongly recommended that you are proficient at the improved beginner's method: some skills for F2L, 4LLL algorithms, and having certain fluency when solving. An average solving time between 20s and 30s is preferred.

A normal studying sequence is: Cross → PLL → F2L → OLL → Advanced

Here is an outline for each part of CFOP:

Content	Learning Period	Skills/Algorithms	Suggestions
1.2 Improved Beginner's	60s → 25s	Free pair, Keyhole 4LLL Algorithms	Important to improve F2L fluency
2.1 Cross (1)	25s → 18s	Simultaneous solving Double base color	Practice seriously in every solve
2.2 PLL	22s → 18s	PLL Algorithms	Memorize in good order
2.1 F2L (1)	20s → 12s	Standard F2L Wrong slot, Back slot PSF2L, other skills	Treat it as a long-term learning
2.2 OLL	18s → 14s	OLL Algorithms	Memorize in good order
3.1 F2L (2) **3.2** LS (basic)	12s → 9s	Cross planning, XCross MAF2L, NSF2L Lookahead, LS Skills	Gradually absorb the advanced content
3.2 OLS/EOLS **3.3** VH/ZB	Under 9s	Advanced LS skills COLL/2GLL algorithms	Personalized long-term learning and practicing

2



*Welcome to the world of speedcubing! This chapter focuses on the intermediate **CFOP Method**, the most widely used system for solving the 3×3×3 cube. You will learn how to improve your solving speed by learning how to build an efficient F2L solving system, recognize patterns, and use algorithms smoothly. This chapter also introduces CFCE and the Roux Method.*

Prerequisite: Strong mastery of the improved beginner's method; sub 30s in 3×3×3 speedcubing.

2.1 First Two Layers (1)

The **CFOP Method** focuses heavily on the first two layers because they are flexible and variable. F2L skills often distinguish advanced cubers from the intermediate cubers, and there is a substantial room to improve in F2L for most cubers. F2L algorithms are helpful, but learning F2L emphasizes understanding the cube's mechanics rather than simply memorizing algorithms.

The first part focuses on:

- Solving the cross quickly
- Solving standard F2L cases efficiently
- Handling non-standard F2L cases
- Improving your F2L solving skills

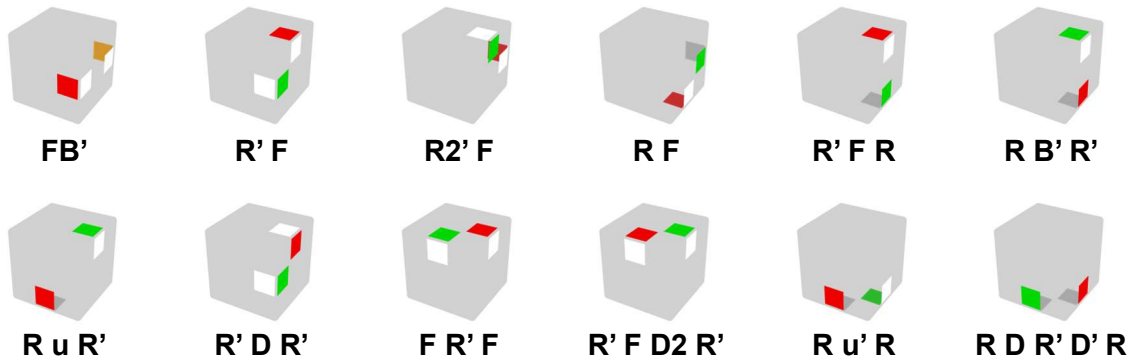
Cross Skills

The cross is the easiest part of solving the Rubik's Cube, but it can also be the most challenging since there are no algorithms to follow. To improve your cross speed, aim to solve it in fewer than 8 moves. This requires discovering the optimal solution on your own, but here are some suggestions:

1. If you find it difficult to solve multiple edges simultaneously, consider these techniques:
   (1) Move another edge to the middle layer while completing an edge.
   (2) Turn the D layer (where the cross is) to position the edge correctly. Remember the order of the four edges: red, blue, orange, and green (clockwise) when solving a white cross.
2. Memorize the positions of the four edges during the inspection and plan your moves to complete the cross blindfolded while practicing.
3. If the white cross proves too difficult, try using yellow as your base color. Practicing with the yellow cross will help improve your two-color cross-solving skills. Remember that the order of the four colors is reversed.

Your goal is to complete the cross in an average of 2 seconds. Remember, practice is the key. Don't be impatient—keep practicing, and you'll improve over time.

Completing 2 edges simultaneously



Standard F2L

As mentioned, while there are algorithms and rules for solving F2L, mastering advanced techniques and developing your skills become more important as you progress. There are 22 F2L “algorithms” (41 counting mirrored cases) introduced in this section. They are numbered so that it is easier to make connections between cases.

A standard F2L case occurs when an edge and corner pair are in the U layer. They are usually easy to recognize. The eventual goal in these situations is to convert them into a free pair. The bolded part of an algorithm is how you convert it into a free pair.

Free Pair

Connected (5+)
U R U’ R’

Connected (5-)
U’ L’ U L

Separated (9+)
R U R’

Separated (9-)
L’ U’ L

We group different cases by the orientation of the white corner or the colors on the top layer. The color patterns work the same way if you are using yellow as your base color.

Top White, Separated

1. Match the edge with the center and hide it in another slot.
2. Connect the corner with the edge to create a connected free pair.

1+
U2 R U R’ U R U’ R’

1-
U2 L’ U’ L U’ L’ U L

2+
U R U2’ R’ U R U’ R’

2-
U’ L’ U2 L U’ L’ U L

Top White, Connected

The connected cases are usually harder to deal with. It is a good idea to just memorize these algorithms for beginners.

3+
R U2 R’ U’ R U R’

3-
L’ U2’ L U L’ U’ L

4+
U R U’ R’ U’ (→ 17+)
R U’ R’ U R U’ R’

4-
U’ L’ U L U (→ 17-)
L’ U L U’ L’ U L

White Rotated, Top Same, Separated

1. Hide the corner in the correct location.
2. Place the edge next to the corner to create a connected free pair.

6+

U' R U R' U2' R U' R'

6-

U L' U' L U2 L' U L

7+

U' R U2' R' U2 R U' R'

7-

U L' U2 L U2' L' U L

White Rotated, Top Same, Connected

8+

R U R' U2' (→ 17+) **R U' R'** U R U' R'

8-

L' U' L U2 (→ 17+) **L' U L** U' L' U L

White Rotated, Top Different

1. Hide the corner in the correct location.
2. Place the edge 1 block away from the corner to create a separated free pair.

10+

U' R U R' U R U R'

10-

U L' U' L U' L U' L'

11+

U' R U' R' U R U R'

11-

U L' U L U' L' U' L

White Rotated, Top Different, Connected

12+

R U' R' U (→ 15+) **R U' R'** U2 R U' R'

12-

L' U L U' (→ 15-) **L' U L** U2' L' U L

Note: The right-hand cases are labeled "+", and the left-hand cases are labeled "-".

Non-Standard F2L

We call a pair of an edge and corner "non-standard" if one or both pieces are in the slot. Non-standard F2L cases occur frequently and can be more challenging to recognize, especially when they are in the wrong slot. However, the idea is still converting them into free pairs. We will first look at cases where the edge/corner is in the correct slot, which count for the 22 algorithms.

Edge in Correct Slot

15+

U' R U' R' U2 R U' R'

15-

U L' U L U2' L' U L

13

(U R U' R')2 U R U' R'

16+

U' R U R' U F' U' F

16-

U F' U' F U' R U R'

14

R U' R' U y' R' U R

Corner in Correct Slot

17+

R U' R' U R U' R'

17-

L' U L U' L' U L

19+

U' L' U L y' U R U' R'

18+

R U R' U' R U R'

18-

L' U' L U L' U' L

19-

U R U' R' y U' L' U L

Edge and Corner in Correct Slot

20+

R U' R' (→ 6+) **U' R U R'** U2' R U' R'

20-

L' U L (→ 6-) **U L' U' L** U2 L' U L

21+

R U' R' (→ 10-) **y**
U L' U' L U' L' U' L

21-

L' U L (→ 10+) **y'**
U' R U R' U R U R'

22

R U' R' (→ 7-) **d R' U2 R** U2' R' U R
R' F R F' (→ 12+) **R U' R' U** (→ 15+)
R U' R' U2 R U' R'

Edge in Wrong Slot

It is also important to deal with cases where one piece of a pair is in a wrong slot. The following instructions explain how to recognize them. Important notes:

1. The patterns involving color order only apply if you are using white base. You need to reverse the order if you are using yellow base.
2. For the two cases on the left, you can use the same principles for the mirrored cases.

U' R U' R'

(1) Color order (left to right): RG, GO, OB, or BR

(2) When the edge is in the adjacent slot, it doesn't align with either of the centers.

(U R U' R')2

(1) The pair isn't aligned.

(2) Color order (left to right): RG, GO, OB, or BR

U' R U R'

(1) Color order (left to right): GR, RB, BO, or OG

(2) When the edge is in the adjacent slot, it aligns with one of the centers.

R U' R'

(1) The pair is aligned.

(2) Color order (left to right): GR, RB, BO, or OG

Corner in Wrong Slot

For the two cases on the left, you can use the same principles for the mirrored cases. These patterns apply for both white base and yellow base.

R U' R'

(1) The side colors of the edge and the corner are different.

(2) The side color of the corner and the top color of the edge are the same.

U R U' R'

Match the side color of the edge and corner in the same face.

U' R U R'

(1) The side colors of the edge and the corner are the same.

(2) The side color of the corner and the top color of the edge are different.

U' L' U L

Match the side color of the edge and corner in the same face.

F2L Skills

Since F2L is very flexible, practicing and developing your skills are as important as learning algorithms. Here are some suggestions and useful skills:

1. Focus on understanding how to pair the corner and edge, and **memorize the methods** rather than just the algorithms. With practice, recognition will become easier.
2. **Reduce whole-cube rotations**. You can try finishing a case when your aim slot is BR or BL once you are familiar with the solving methods. Refer to the examples below. We can also have back options for non-standard F2L cases, but they are harder to recognize.
3. When you have multiple choices for a pair, follow these **priorities**: (1) Solve the easiest pair first; (2) Solve the pair in the back slots first; (3) Solve the first pair you see.
4. The **Keyhole Method** is a valuable skill. An advanced technique, **Pseudo Slotting** (PSF2L), is used for cases where a corner and edge not in a pair are already solved, but it can be hard to recognize. Be careful when using this skill. See the simple examples below.
5. **Improve recognition** and **avoid long pauses**. Try finding the next pair to solve while completing the current one. This is an advanced skill called “lookahead”. It will be discussed in F2L part (2).

Back Slot Solutions

U2 R’ U’ R U’ R’ U R

U’ R’ U R U R’ U R U’ R’ U R

U’ R’ U R

U’ R’ U R

R’ U R d’ R U R’ U R U R’

U’ L U L’ U L U L’

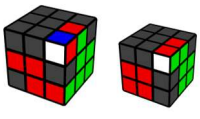
L U2 L’ U’ L U L’

U’ L U L’ U2 L U’ L’

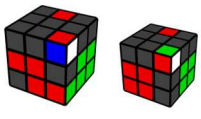
L U L’ U’ L U L’

L U’ L’ U L U2’ L’ U L U’ L’

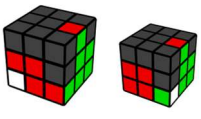
Pseudo Slotting



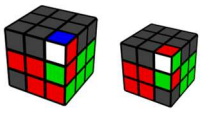
D U R U’ R’ D’



D R U R’ D’



D R U R’ U’ R U R’ D’



D U’ R U R’ U F’ U’ F D’

2.2 OLL and PLL Algorithms

The last layer of the CFOP Method offers the advantages of a low move count, simple recognition, and medium difficulty of memorization. It involves 57 **OLL algorithms** and 21 **PLL algorithms**, and many of them are covered in the improved beginner's method. This tutorial covers both right-handed and left-handed algorithms. Additionally, it provides special CFOP algorithms designed for one-handed solving.

Right-Handed OLL Algorithms (57)

Recognition and memorization

First, judge the shape facing up, then examine the lines, bars, and lights on the side faces.

It is recommended to learn according to the grouping order. While learning and memorizing, pay attention to reverse algorithms and similar patterns.

OCLL

S+
R U R' U R U2' R'

S-
R' U' R U' R' U2 R

H
(R U R' U) (R U' R' U) R U2' R'

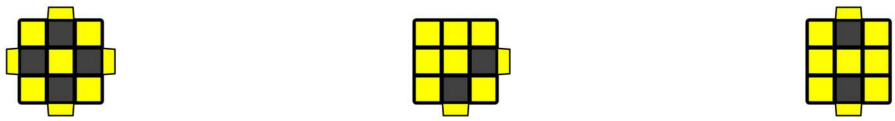
π
R U2' R2' U' R2 U' R2' U2' R

T
(r U R' U') (r' F R F')

L
F' (r U R' U') r' F R

U
R2 D (R' U2 R D') R' U2 R'

OELL

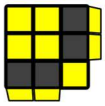


(r U R' U') M2' (U R U' R') U' M'

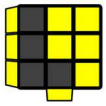
(r U R' U') (r' R U R U' R')

(R U R' U') (M' U R U' r')

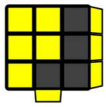
Sexy Variants



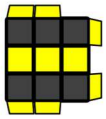
(F R' F' R) (U R U' R')
F (R U' R' U') (R U R' F')



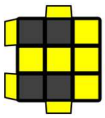
F' (U' L' U L) F
[U2] f' (L' U' L U) f



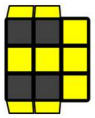
F (U R U' R') F'
[U2] f (R U R' U') f'



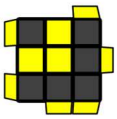
F (U R U' R')2 F'
[U2] f (R U R' U')2 f'



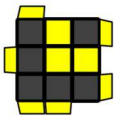
F (R U R' U') F'



(R U R' U') (R' F R F')

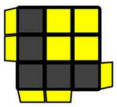


F (R U R' U')2 F'

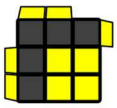


R' U' (R' F R F')2 U R

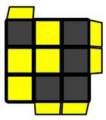
Sune Variants



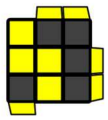
r U2' R' U' R U' r'



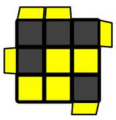
r' U2' R U R' U r



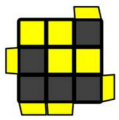
r U R' U R U2' r'



r' U' R U' R' U2 r

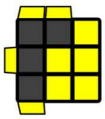


r' (R2 U R' U) (R U2' R' U) M'

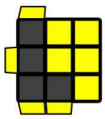


M' (R' U' R U') (R' U2 R U') r' R

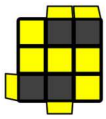
9-mover



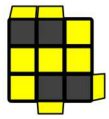
S (R U R' U') (R' F R f')



R' U' F (U R U' R') F' R

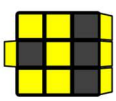


R' F (R U R' U') F' U R

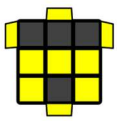


L F' (L' U' L U) F U' L'

C

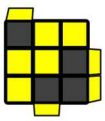


R' U' (R' F R F') U R

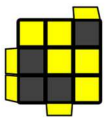


(R U R2' U') (R' F R U R U' F')

W



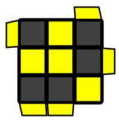
(R U R' U) (R U' R' U') (R' F R F')



(L' U' L U') (L' U L U) (L F' L' F)

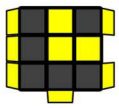
Y

(R U R' U) (R' F R F') R U2' R'

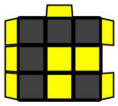


(R U R' U') (R' F R2 U R' U' F')

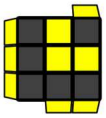
V



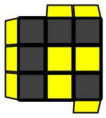
(r U R' U) (R U' R' U)
R U2' r'



(r' U' R U') (R' U R U')
R' U2 r

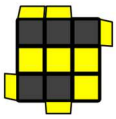


r U' r2' U r2 U r2'
U' r

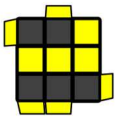


r' U r2 U' r2' U'
r2 U r'

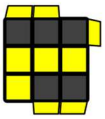
J



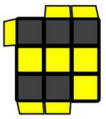
r' U' r (R' U' R U) r'
U r



r U r' (R U R' U') r
U' r'

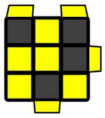


F U (R U2 R' U') (R U
R' F')

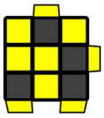


R' F (R U R' F') (R
F U' F')

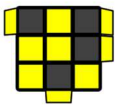
Glider



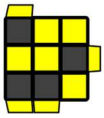
(R U R' U R U2' R')
(F R U R' U' F')



(R' U' R U' R' U2 R)
(F R U R' U' F')

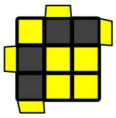


F U (R U2 R' U')2 F'



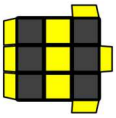
(R U R' U') (R U' R')
(F' U' F) (R U R')

Turtle

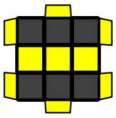


(R U2' R2' F R F')
R U2' R'

I



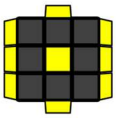
(R' F' U' F U')
(R U R' U R)



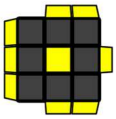
r U r' (U R U' R')2
r U' r'

(R' F R U R U' R2' F')
(R2 U' R' U R U R')

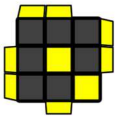
Dot



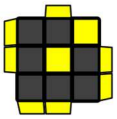
(R U2' R2' F R F')
U2 (R' F R F')



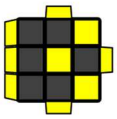
F (R U R' U') S
(R U R' U') f'



(f R U R' U' f') U'
(F R U R' U' F')

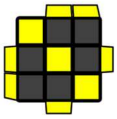


(f R U R' U' f') U
(F R U R' U' F')



(R U2' R2' F R F') U2
(M' U R U' r')

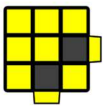
r' R U (R U R' U') r
(R2' F R F')



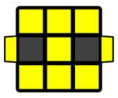
(R U R' U) (R' F R F') U2
(R' F R F')

S-Style Algorithms

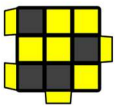
The OLL algorithms above are the most popular ones. Here are some S style algorithms if you prefer them. The yellow ones are recommended.



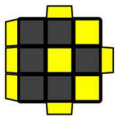
(R' F R S)
(R' F' R S')



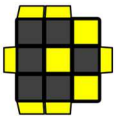
(R U' R' S')
(R U R' S)



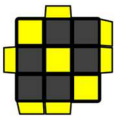
S'F (R U R' U')
F' U S



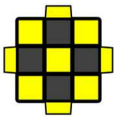
FS' (R U' R') S
(R U2 R' U' F')



(S' R U R' S) U' (R' F R F')



(F R' F' R) U (S' R U' R' S)



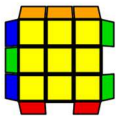
(S' R U R' S) U' (M' U R U' r')

Right-Handed PLL Algorithms (21)

Recognition and memorization

The pictures show the key patterns for each case, which can include lines, bars, lights, or opposite colors. There are multiple algorithm options available, the bolded ones are recommended. [X] implies the initial orientation or the final **AUF** (Adjustment of U face). Probabilities are indicated. It is recommended to start with adjacent corner swaps and algorithms with high probability.

EPLL

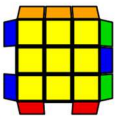


1/18 Ua

M2' U' M U2 M' U' M2'

[U2] (R' U R' U') R3 U' (R' U R U R2')

R2' U (R U R' U') R3 U' R' U R'

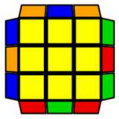


1/18 Ub

M2' U M U2 M' U M2'

[U2] (R U R' U) (R' U' R2 U') (R' U R' U R)

[U'] R2 U' S' U2' S U' R2

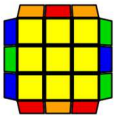


1/36 Z

(M2' U)2 M' U2 M2' U2 M'

[U'] M' U (M2' U)2 M' U2 M2'

S M2' S' u M2' u' M2'

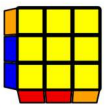


1/72 H

M2' U M2' U2 M2' U M2'

(R2 U2' R U2 R2') U2 (R2 U2' R U2 R2')

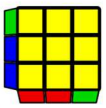
CPLL



1/18 Aa (adj)

x (R' U R') D2 (R U' R') D2 R2 x'

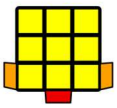
[U] x' R2 D2 (R' U' R) D2 R' U R' x



1/18 Ab (adj)

x R2' D2 (R U R') D2 (R U' R) x'

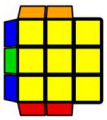
[U] x' (R U' R) D2 (R' U R) D2 R2' x



1/36 E (opp)

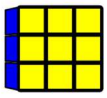
x' (R U' R' D) (R U R' D') (R U R' D) (R U' R' D') x

Adjacent Corner Swap (2C2E)



1/18 T

(R U R' U') (R' F R2 U' R' U') (R U R' F')

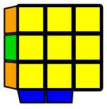


1/18 F

R' U' F' (R U R' U') (R' F R2 U' R' U') (R U R' U R)

1/18 J

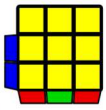
(R U R' F') (R U R' U') (R' F R2 U' R') [U']



1/18 Ra

(R U' R' U') (R U R D) (R' U' R D') (R' U2 R') [U']

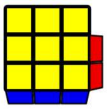
(R U R' F') (R U2 R' U2') (R' F R U) (R U2' R') [U']



1/18 Rb

(R' U2 R' D') (R U' R' D) (R U R U' R' U' R) [U']

(R' U2 R U2') (R' F R U R' U') (R' F' R2) [U']

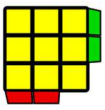


1/18 L

(R' U L' U2) (R U' R' U2) R L [U']

[U'] (L' U' L F) (L' U' L U) (L F' L2' U L) [U2']

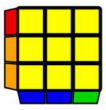
Opposite Corner Swap (2C2E)



1/18 Y

F (R U' R' U' R U R' F') (R U R' U' R' F R F')

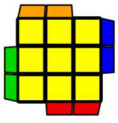
F (R' F R2 U' R' U') (R U R' F') (R U R' U') F'



1/18 V

[U] R U' (R U R' D R D' R U'D) (R2' U R2 D' R2')

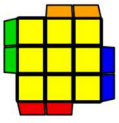
(R' U R U') R' f' U' (R U2 R' U' R U' R') f R



1/72 Na

(R U R' U R U R' F' R U R' U') (R' F R2 U' R' U2 R U' R')

z (U R' D R2 U') (R UD' R' D R2 U' R D') z' [U']

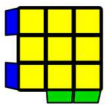


1/72 Nb

(R' U R U' R' F' U' F R U R' F) (R' F' R U' R)

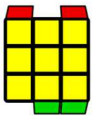
(R' U R U' R' F' U' F) (R U R' U') R U' f R f'

G Perm (3C3E)



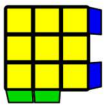
1/18 Ga

D' (R2 U R' U R' U' R U') (R2 U'D R' U R) [U]



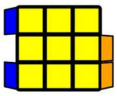
1/18 Gb

(F' U' F R2 u) (R' U R U' R u' R2')



1/18 Gc

(R2' F2 R U2 R U2') (R' F R U R' U') (R' F R2)



1/18 Gd

D' (R U R' U'D) (R2 U' R U') (R' U R' U R2) [U]

Left-Handed OLL Algorithms

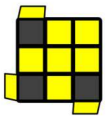
Recognition and memorization

First, judge the shape facing up, then examine the lines, bars, and lights on the side faces.

Line Bar Light Dot

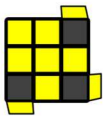
It is recommended to learn according to the grouping order. While learning and memorizing, pay attention to reverse algorithms and similar patterns.

OCLL



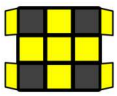
S+

L U L' U L U2' L'



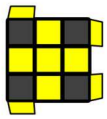
S-

L' U' L U' L' U2 L



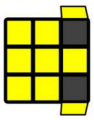
H

(L' U' L U') (L' U L U') L' U2 L



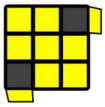
π

L' U2 L2 U L2' U L2 U2 L'



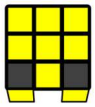
T

(l' U' L U) (l F' L' F)



L

F (l' U' L U) l F' L'

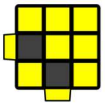


U

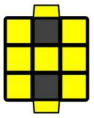
L2' D' (L U2' L' D) L U2' L

OELL

(I' U' L U) M2'
(U' L' U L) U M'

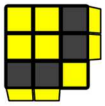


(I' U' L U) (I L' U' L' U L)

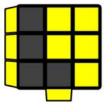


(L' U' L U) (M' U' L' U I)

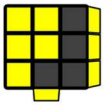
Sexy Variants



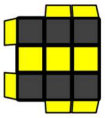
(F' L F L') (U' L' U L)
F' (L' U L U) (L' U' L F)



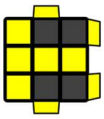
F' (U' L' U L) F
[U2] f' (L' U' L U) f



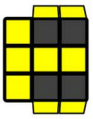
F (U R U' R') F'
[U2] f (R U R' U') f'



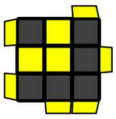
F' (U' L' U L)2 F
[U2] f' (L' U' L U)2 f'



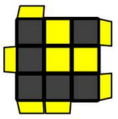
F' (L' U' L U) F



(L' U' L U) (L F' L' F)

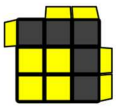


L U (L F' L' F)2
U' L'

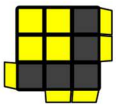


F' (L' U' L U)2 F

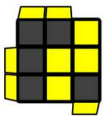
Sune Variants



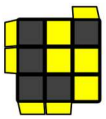
I' U2 L U L' U I



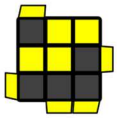
I U2 L' U' L U' I'



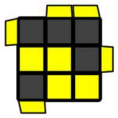
I' U' L U' L' U2 I



I U L' U L U2' I'

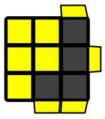


M' (L U L' U) (L U2' L' U) I L'

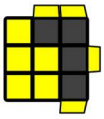


I (L2' U' L U') (L' U2 L U') M'

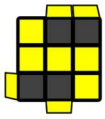
9-mover



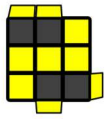
L U F' (U' L' U L)
F L'



S' (L' U' L U)
(L F' L' F)

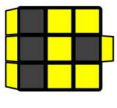


R' F (R U R' U')
F' U R

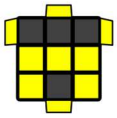


L F' (L' U' L U)
F U' L'

C

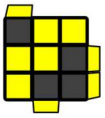


L U (L F' L' F) U' L'

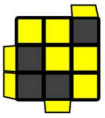


(L' U' L2 U) (L F' L' U' L' U F)

W

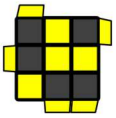


(R U R' U) (R U' R' U') (R' F R F')

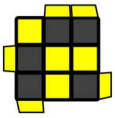


(L' U' L U') (L' U L U) (L F' L' F)

Y

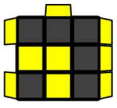


(L' U' L U) (L F' L2' U' L U F)

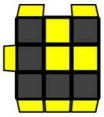


(L' U' L U') (L F' L' F) L' U2 L

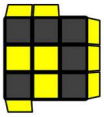
V



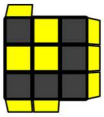
(I' U2' L U) (L' U' L U) L' U I



I U (L' U L U') (L' U L U2') I'

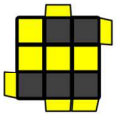


I U' I2' U I2 U I2' U' I

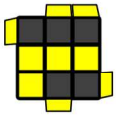


I' U I2 U' I2' U' I2 U I'

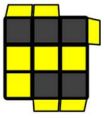
J



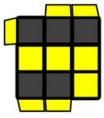
I' U' I (L' U' L U) I' U I



I U I' (L U L' U') I U' I'

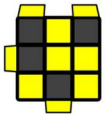


L F' (L' U' L F) (L' F' U F)

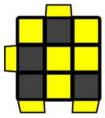


F' U' (L' U2' L U) (L' U' L F)

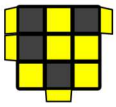
Glider



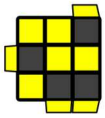
(L' U' L U' L' U2 L) (F' L' U' L U F)



(L U L' U L U2' L') (F' L' U' L U F)

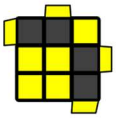


F' U' (L' U2' L U)2 F

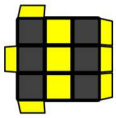


(L' U' L U) (L' U L) (F U F') (L' U' L)

Turtle

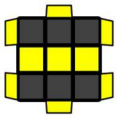


(L' U2 L2 F' L' F) L' U2 L

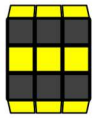


(L F U F' U) (L' U' L U' L')

I

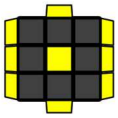


I' U' I (U' L' U L)2 I' U I

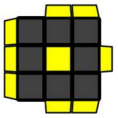


(L F' L' U' L' U L2 F) (L2' U L U' L' U' L)

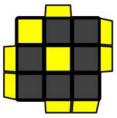
Dot



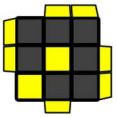
(L' U2 L2 F' L' F) U2
(L F' L' F)



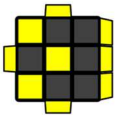
F' (L' U' L U) S'
(L' U' L U) f



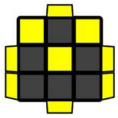
(f' L' U' L U f) U
(F' L' U' L U F)



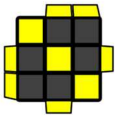
(f' L' U' L U f) U'
(F' L' U' L U F)



(L' U2 L2 F' L' F) U2'
(M' U' L' U l)



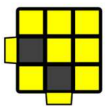
l L' U' (L' U' L U) l'
(L2 F' L' F)



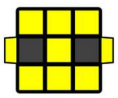
(L' U' L U') (L F' L' F) U2'
(L F' L' F)

S-Style Algorithms

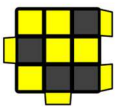
The OLL algorithms above are the most popular ones. Here are some S style algorithms if you prefer them. The yellow ones are recommended.



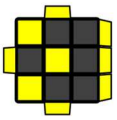
(L F' L' S')
(L F L' S)



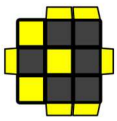
(L' U L S)
(L' U' L S')



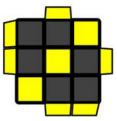
S F' (L' U' L U)
F U' S'



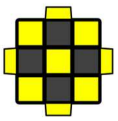
F' S (L' U L) S'
(L' U2' L U F)



(S L' U' L S') U (L F' L' F)



(F' L F L') U' (S L' U L S')



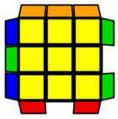
(S L' U' L S') U (M' U' L' U L)

Left-Handed PLL Algorithms (21)

Recognition and memorization

The pictures show the key patterns for each case, which can include lines, bars, lights, or opposite colors. There are multiple algorithm options available, the bolded ones are recommended. [X] implies the initial orientation or the final **AUF** (Adjustment of U face). Probabilities are indicated. It is recommended to start with adjacent corner swaps and algorithms with high probability.

EPLL

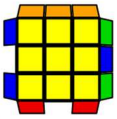


1/18 Ua

M2’ U’ M U2’ M’ U’ M2’

[U2’] (L’ U’ L U’) (L U L2’ U) (L U’ L U L’)

[U] L2’ U S U2 S’ U L2’

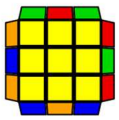


1/18 Ub

M2’ U M U2’ M’ U M2’

[U2’] (L U’ L U) L’3 U (L U’ L’ U’ L2)

L2 U’ (L’ U’ L U) L’3 U L U’ L

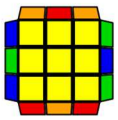


1/36 Z

(M2’ U’)2 M’ U2’ M2’ U2’ M’

[U] M’ U’ (M2’ U’)2 M’ U2’ M2’

S’ M2’ S u’ M2’ u M2’

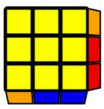


1/72 H

M2’ U’ M2’ U2’ M2’ U’ M2’

(L2’ U2 L’ U2’ L2) U2’ (L2’ U2 L’ U2’ L2)

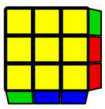
CPLL



1/18 Aa (adj)

x L2 D2’ (L’ U’ L) D2’ (L’ U L’) x’

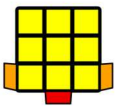
[U’] x’ (L’ U L’) D2’ (L U’ L’) D2’ L2 x



1/18 Ab (adj)

x (L U’ L) D2’ (L’ U L) D2’ L2’ x’

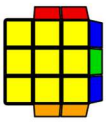
[U’] x’ L2’ D2’ (L U L’) D2’ L U’ L x



1/36 E (opp)

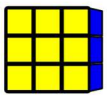
x’ (L’ U L D’) (L’ U’ L D) (L’ U’ L D’) (L’ U L D) x

Adjacent Corner Swap (2C2E)



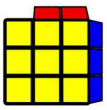
1/18 T

(L’ U’ L U) (L F’ L2’ U L U) (L’ U’ L F)



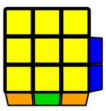
1/18 F

L U F (L’ U’ L U) (L F’ L2’ U L U) (L’ U’ L U’ L’)



1/18 L

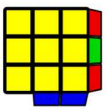
(L’ U’ L F) (L’ U’ L U) (L F’ L2’ U L) [U]



1/18 Ra

(L U2’ L D) (L’ U L D’) (L’ U’ L’ U L U L’) [U]

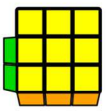
(L U2’ L’ U2) (L F’ L’ U’ L U) (L F L2’) [U]



1/18 Rb

(L’ U L U) (L’ U’ L’ D’) (L U L’ D) (L U2’ L) [U]

(L’ U’ L F) (L’ U2’ L U2) (L F’ L’ U’) (L’ U2 L) [U]

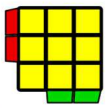


1/18 J

(L U’ R U2’) (L’ U L U2’) L’ R

[U] (R U R’ F’) (R U R’ U’) (R’ F R2 U’ R’) [U2]

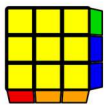
Opposite Corner Swap (2C2E)



1/18 Y

F’ (L’ U L U L’ U’ L F)
(L’ U’ L U L F’ L’ F)

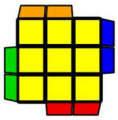
F’ (L F’ L2’ U L U)
(L’ U’ L F) (L’ U’ L U) F



1/18 V

[U’] L’ U (L’ U’ L D’ L’ D L’ UD’)
(L2 U’ L2’ D L2)

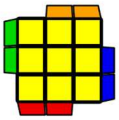
(L U’ L’ U) L f U (L’ U2’ L U L’ U L) f’ L’



1/72 Na

(L U’ L’ U L F U F’ L’ U’ L F’)
(L F L’ U L’)

(L U’ L’ U L F U F’)
(L’ U’ L U) L’ U f’ L’ f

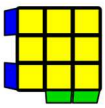


1/72 Nb

(L’ U’ L U’ L’ U’ L F L’ U’ L U)
(L F’ L2’ U L U2’ L’ U L)

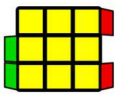
z’ (U’ L D’ L2’ U) (L’ U’D L D’ L2’ U L’ D) z
[U]

G Perm (3C3E)



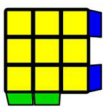
1/18 Ga

(L2 F2’ L’ U2’ L’ U2) (L F’ L’ U’ L U)
(L F’ L2’)



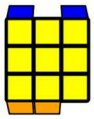
1/18 Gb

D (L’ U’ L UD’) (L2’ U L’ U) (L U’ L U’ L2’)
[U’]



1/18 Gc

D (L2’ U’ L U’ L U L’ U)
(L2’ UD’ L U’ L’) [U’]



1/18 Gd

(F U F’ L2’ u’) (L U’ L’ U L’ u L2)

One-Handed Algorithms

3×3×3 One-handed is a WCA event in which solvers use only one hand to solve the cube. It requires different turning techniques, finger control, and suitable algorithms. Most solvers use their non-dominant hand (the left hand for right-handed solvers), because it makes R moves easier to perform with the ring finger. The key idea is to use fewer move types, or generators, so the cube can be turned more smoothly. For F2L, it is often better to use mainly R, U, and r moves, while for OLL and PLL, wide turns and cube rotations can make algorithms easier to execute one-handed.

F2L (6)

U2' R U R' F' U' F

U2' R2 U2' R' U' R U' R2

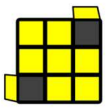
(R U' R' U) (r U' R' U R U r')

U (R U' R' U') (R U R' U') R U R'

(R U' R' U') (R U' R' U) R U R'

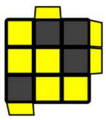
(R2 U2' R' U') (R U' R' U2' R')

OLL (18)

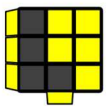


F (R U' R' U') (R U2' R' U') F'

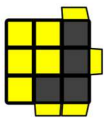
R U R' U' f' z x R' U R U'D x'



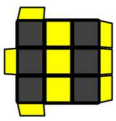
(R' U' R U') (R' U R U) (R f' U' f)



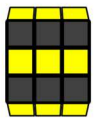
z F' (R' U' R U) F z'



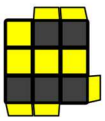
z u' (R2 U R U2) (R D R' U R) x'



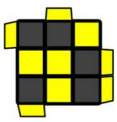
(R U R' U R d') (R U' R' F')



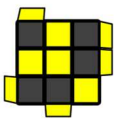
(r U2' R' U') R2 (r' U R' U') r U' r'



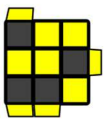
(R U R' U) (r' U' R2 U' R2' U2' r)



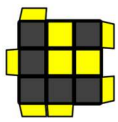
r' R2 U2' (R' U' R U') (R' U' r R')



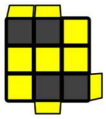
r R2' U2' (R U R' U) (R U r' R)



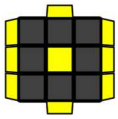
(r U R' U') r' R L' (U R U' R') r x'



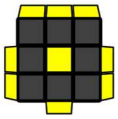
z F' (U' R' U R)2 F z'



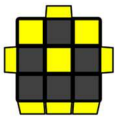
r U' r' U' r z x' U R U' F'



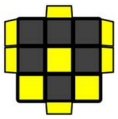
(r' U' R U' R' U2' r) U
(r U2' R' U' R U' r')



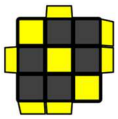
(r U r' U2') (R U2' R' U2)
r U' r'



(r U R' U R U2' r2)
(U' R U' R' U2' r)

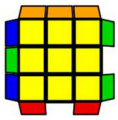


(r U2' R' U' R U')
(r2 U2' R U R' U r)

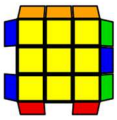


r U' z U' R x' U2' R d'
(R U' R' F')

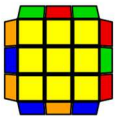
PLL (12)



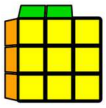
U+
z U' (R U' R' U') (R' U' R U)
R U2' z'



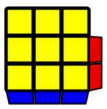
U-
(R U' R U) (R U R U')
R' U' R2



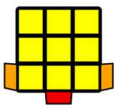
Z
(R' U' R U') (R U R U')
(R' U R U) R2 U' R'



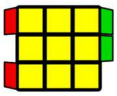
J
(R U2' R' U') R U2'
(L' U R' U' r) x'



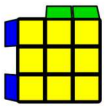
L
(R' U L' U2) (R U' R' U2)
R r x'



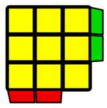
E
(L U' R' D) (R U R' u2')
(R' U R D) R' U' r x'



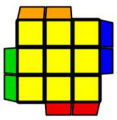
Gb
R' U' R y (R2 u R' U)
(R U' R u') R2'



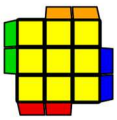
Gc
(R2 u' R U') (R U R' u R2)
y R U' R'



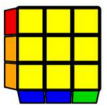
Y
R2 U' (R' U R U') x' U' z'
U' (R U' R' U') z U R x



Na
z U R' D (R2 U' R UD')
R' D (R2 U' R D') z'



Nb
z U' R D' (R2 U R' U'D)
R D' (R2 U R' D) z'



V
R' U R U' x' U (R U2' R' U')
(R U' R' U2') (R U R' U') x

2.3 CFCE & Roux

There are many speedcubing methods beyond CFOP. This tutorial introduces two intermediate alternatives: CFCE and Roux. CFCE keeps the same F2L structure as CFOP but uses a different last-layer approach, while Roux has a very different block-building structure. Despite this difference, CFCE and Roux share some similarities: Both methods are corner-first regarding the last layer; CLL/CMLL algorithms are closely related.

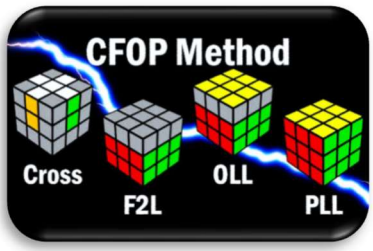
Comparing the Methods

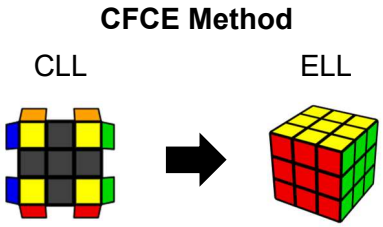
CFOP Method – The most popular method in contemporary speedcubing. It is the easiest to use for recognizing structures among these methods and supports advanced learning.

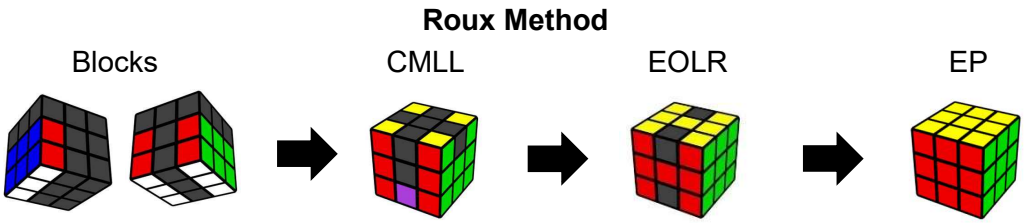
CFCE Method – Developed concurrently with CFOP. It requires the fewest moves among these three methods but can be challenging to recognize structures. The CLL (Corner Last Layer) part has similarities with the 2×2×2 CLL algorithms.

Roux Method – Developed slightly later than CFOP; based on block building rather than solving F2L pairs. Roux uses more intuitive solving, fewer rotations, and finishes with a unique last-six-edges step (which can be complicated).

	CFOP Method	CFCE Method	Roux Method
1st Solving Part	Cross, F2L	Cross, F2L	Left & Right Blocks
2nd Solving Part	OLL, PLL	CLL, ELL	CMLL, LSE
Number of Moves	Good	Great	Great
Recognition	Fairly easy	Medium-hard	Hard
Memorization	Medium	Medium	Medium-easy







CFCE Method

CFCE stands for Cross, F2L, **CLL**, and **ELL**. The idea of solving the last layer is straightforward: Solve the corners and then solve the edges. The CFCE Method is much less popular than the CFOP Method mostly because its patterns are less intuitive.

CLL Algorithms (42)

Recognition and Memorization: Colors are crucial for recognition in CLL. You should be very familiar with opposite and adjacent colors. S+ and S- cases are generally more challenging to recognize because you need to inspect the colors on the side. Many of the algorithms are the same as OLL algorithms, so it would be easy to memorize them if you already know OLL algorithms.

Note: The small letters beside the image indicate which two corners need to be swapped. X = Corners permuted. D = Diagonal. B/F/L/R = the back/front/left/right pair. Light blue: The same algorithms as the one in 2×2×2 CLL.

O

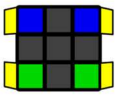
A

(R U R' F') (R U R' U') (R' F R2 U' R')

D

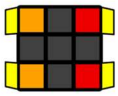
R2 D (r' U r D') R2' U' (F' U' F)

H



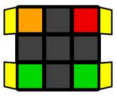
X

(R U R' U)
(R U' R' U) R U2' R'



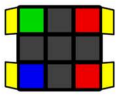
D

U F (U R U' R')3 F'



B

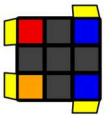
R U2' (R2' F R F')
U2' (R' F R F')



R

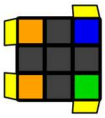
(F R' F' R) U2' F2
(r U r' F)

π



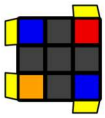
X

R U2' R2' U' R2 U'
R2' U2' R



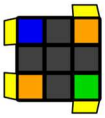
D

U' R' U' (R' F R F')
R U' R' U2 R



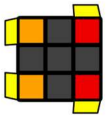
B

U (F R' F' R) U2
(R U' R' U) R U2' R'



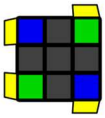
F

(R U2' R' U2)
(R' F R2 U R' U' F')



L

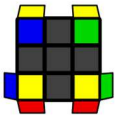
U2 (r' F R F') (r U' R' U')
R U' R'



R

U' F (R U' R' U')
(R U R' U) (R U' R' F')

U



X

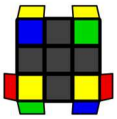
(R’ U’ R U’ R’ U2’)
(R2 U R’ U R U2’ R’)

D

U’ F (R U R’ U’) F’

B

U’ F R2 D (R’ U R D’)
R2’ U’ F’



F

U2 F (R U’ R’ U) (R U R’ U)
(R U’ R’ F’)

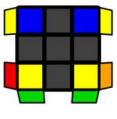
L

R2’ D’ (R U2 R’ D) R U2 R

R

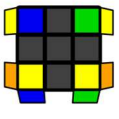
U2 R2 D (R’ U2 R D’)
R’ U2 R’

T



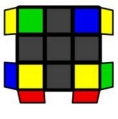
X

F R’ F (R2 U’ R’ U’)
(R U R’ F2)



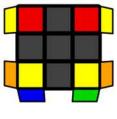
D

U2 r2’ D’ (r U r’ D)
(r2 U’ r’ U’ r)



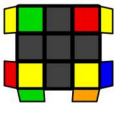
B

(R’ U R U2’) r’ (R’ F R F’) r



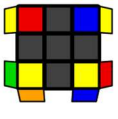
F

F U (R U2 R’ U’)2 F’



L

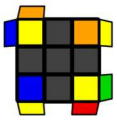
U’ (R U R’ U’) (R’ F R F’)



R

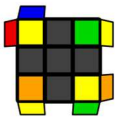
U (L’ U’ L U) (r U’ r’ F)

L



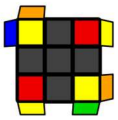
X

U2 (R U R’ U) (R U’ R’ U)2
R U2’ R’



D

R U2’ R2’ F R F’ R U2’ R’

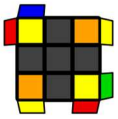


B

(R U2 R D) (R’ U2 R D’)
R2’

F

F (R U’ R’ U’) (R U R’ F’)



L

U’ (R’ U2 R’ D’)
(R U2 R’ D) R2

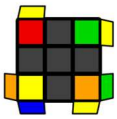
R

(F R’ F’ R) (U R U’ R’)

S+

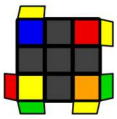
X

R U R' U R U2' R'



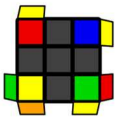
D

U2 (R U R' U) (R' F R F') R U2' R'



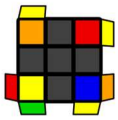
B

(L' U2 L U2') (r U' r' F)



F

U (R' F2 R2 U2' R') F (R U2' R2' F2 R)



L

r U' r' F R' F' R

B

(F R' F' R) (U2 R U2' R')

S-

X

U2 R' U' R U' R' U2 R

D

U R' U' (R U' R' U) (R' F R F') U R

B

(R U2' R' U2) (R' F R F')

F

U' (R' F2 R2 U2' R') F' (R U2' R2' F2 R)

L

(F' r U r') (U2' L' U2 L)

R

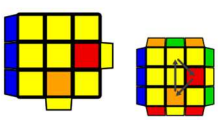
(R' F R F') r U r'

ELL Algorithms (29)

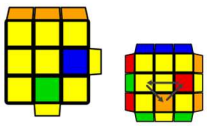
Recognition and Memorization: ELL recognition is relatively easier than CLL, and they are somewhat helpful to CFOP solvers (a chance of 1/27). There are two pictures for each case: the larger one highlights the key points of recognition, the smaller one shows the permutation. Memorizing the initial orientations can be a bit challenging.

Note: Blue algorithms are OELL algorithms; Yellow algorithms are OLL combinations.

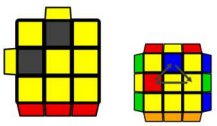
A (Arrow)



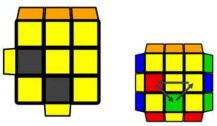
(r U R' U') (M U R U' R')



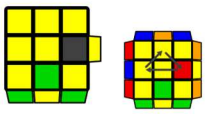
(R' F R S) (R' F' R S')



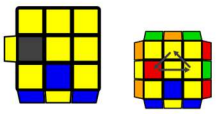
M' U M U2 M' U M



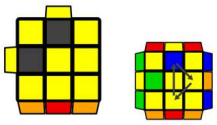
M U' M' U2 M U' M'



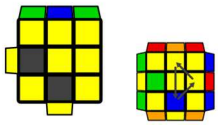
M' U' M2' U'
M U2 M' U' M'



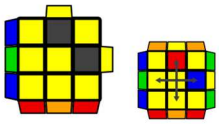
M' U M2' U
M U2 M' U M'



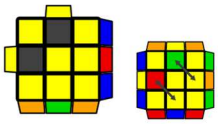
(r U r' U2') (R2 U2' R2'
U' R2 U' M'R') U' r'



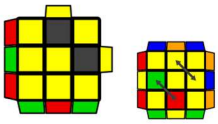
(r' U' r U2') (R2' U2'
R2 U R2' U MR) U r



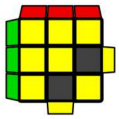
(R' U' R' F R F' U R)
(F R U R' U' F')



(F R U R' U' F')
(R' U' R' F R F' U R)

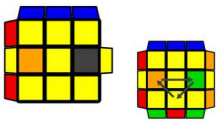


(R' U' F' U F R) U'
(f R U R' U' f')

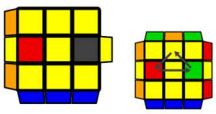


Flip
R' U2' (R2 U R' U')
R' U2' (r U R U' r')

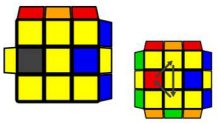
B (Helipad)



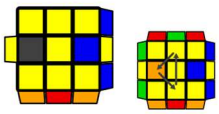
(R U R' S')
(R U' R' S)



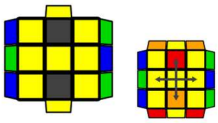
(R U' R' S')
(R U R' S)



r' U' (M'R' U' R2 U')
(R2' U2' R2 U2') r' U r

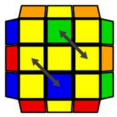


r U (MR U R2' U) (R2
U2' R2' U2') r U' r'

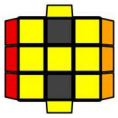


(r U r' R U R' U' r U' r')
(R U R' U R U2' R')

(M' U' M U)
(M' U' M U') M' U2 M



(M' U M U')
(M' U M U) M' U2 M

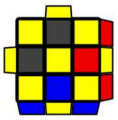


Flip
M' U2 M U (R' F'
R S R' F R S')

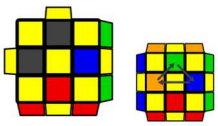
X (Super Flip)

(S' R U R' S U')
(M' U R U' r')

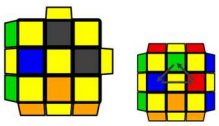
R2' S' R2 U' S2' U'
R2' S' R2



Flip
(M' U2 M U2) M' U M
(U2 M' U2 M)



(f R U R' U' f')
(R' U' R' F R F' U R)



(R' U' F) (R' F' R U R) U2'
(F R U R' U' F')

Roux Method

The **Roux Method** is a 3×3×3 speedcubing method developed by Gilles Roux around 2003. Unlike CFOP and CFCE, which are based on F2L pairs, Roux uses block building. The solve begins by building two 1×2×3 blocks on the left and right sides, then solving the top-layer corners with CMLL, and finally solving the remaining six edges using M and U moves.

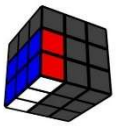
The Roux Method is overall a more intuitive solving method than CFOP and CFCE: It uses fewer algorithms but more direct constructions. This brings significant advantages as well as drawbacks. In Roux, constructing 1×2×3 blocks is more flexible than inserting F2L pairs; it is easier to predict far ahead during inspection; the turning style suits one-handed solving. The disadvantages include a less structured solve, greater uncertainty, and a more complicated LSE stage.

This book does not dive into the Roux Method since it is less popular than CFOP, but Roux can be a good choice if you enjoy intuitive solving or you are stuck in improving your CFOP skills. When learning Roux, focus first on understanding block building. After that, try to tackle CMLL algorithms and advanced EOLR techniques.

First Block (FB)

The goal of First Block is to construct a 1×2×3 block on the left side of the cube (if you are right-handed). This step is mostly by intuition, but there are some common strategies:

(1) Solve one edge on the bottom, then solve two remaining pairs.
(2) Build a 1×2×3 block first, then solve the last pair.
(3) Use an S move to insert the edge on the bottom and make a square.
(4) Solve two 1×1×3 lines using D or u moves.



When practicing, try completing FB blindfolded, and always look for a better solution before starting right away. This would help you in real timed solves.

Second Block (SB)

The goal of Second Block is to construct a 1×2×3 block on the right side of the cube. You should complete the 1×2×2 block on the back first by intuition. Ideally, you should be looking ahead to this step when you are doing the first block.

To make the Roux Method efficient, you must leverage the unique advantage of not solving the middle center rather than doing the same thing as in CFOP. The most important technique is using **wide inserts** involving r or M moves. Since M moves to flip the orientation of the edges, you do not need to rotate the cube at most times.

Learn the following cases by understanding instead of memorizing algorithms. You can apply the same techniques to other slots during FB and SB.

r U r'

r U' r'

R U2 M' U R'

M2' U2 R U' R'

(R U R' U') r U r'

U' M U M' r U r'

r U2' M2' U' r'

R U' R' U' M R U R'

M2 U r U r'

F' U' F

U2 M2 U' r U' r'

U' M2' U2 r U' r'

r U2' r' U' M' r U r'

M U r U' r'

M' (U R U' R')2

U2 r U r' U r U' r'

r U' r' U2 M' r U r'

r U' r' U2 r U r'

M' U (R U2' R')
U R U' R'

U r U2' r' U r U' r'

U R U' M' U R'

U' r U' M U r' U'

R U' R' U2 M U'
r U' r'

r U r' U M' U2
R U R'

R U' R' U2 r U r'

R U' M2' U2 M' U' r'

r U' r' M U r U' r'

r U r' U' M' r U r'

U2 R U M' U R'

r U' r' M U' r U' r'

R U R' U r' U' r'

r2 U2 R' U' R U' R2'

Last Layer Corners (CMLL)

This step is essentially the same as CLL in CFCE Method: Fix both the orientation and the permutation of the corners. You can use exactly the CLL algorithms for CMLL. Don't be distracted by the edges or the centers.

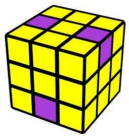
(One could argue that Roux is less efficient than CFCE because leaving the centers and edges unsolved only benefits the block-building/F2L stage, but does not help with solving the top-layer corners, plus it makes the final LSE step more cumbersome.)

Edge Orientation (EO)

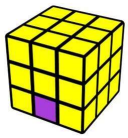
You will use only M and U moves for solving the rest of the cube, the last 6 edges. An edge is oriented if the yellow/white sticker faces up/down, misoriented if the yellow/white sticker is on the F/B face. Since there must be even number of oriented edges, you don't need to inspect the edge behind that you can't see. Focus on inspecting the 5 edges: UB, UL, UR, UF, DF.

Purple edges indicate misoriented edges. Here are 9 basic cases. You can use M moves to adjust if you encounter other cases.

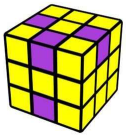
R U' r' U' M' U r U r'



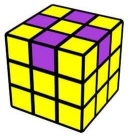
M' U2 M' U2 M U' M'



M' U' M' U M U M'

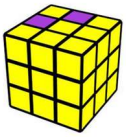


M' U M'

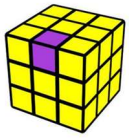


M' U2 M' U2 M' U' M'

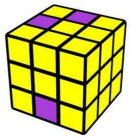
M' U' M U M' U M'



M' U M' U2 M' U' M'



M' U M' U M U M'



U M2' U M' U M'

UL/UR Edge

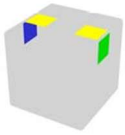
You can either solve intuitively or use algorithms for this step. For intuitive solving, use M2 and U turns (so the edge orientation wouldn't be affected) to move the two edges to spot UB/DF, or UF/DB, combine them, and insert them. Advanced Roux solvers often combine the EO step and the UL/UR step, and this technique (actually a Set of algorithms) is called EOLR.

M' U2 M'

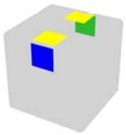
M' U2 M

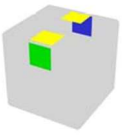
Combined

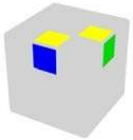
UL/UR Algorithms


Solved

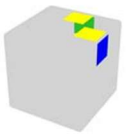
U M2' U2 M2' U

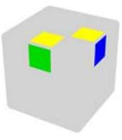

M2' U' M2' U


M2' U M2' U'

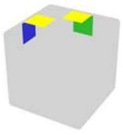

M2' U M' U2 M' U

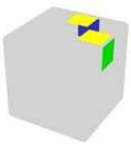
M2' U' M U2 M' U'


M2' U' M' U2 M U'


M2' U M U2 M' U

M2' U' M' U2 M' U'


M2' U M U2 M U


M2 U' M U2 M U'

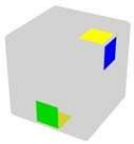
M2' U M' U2 M U

U2 M' U2 M' U M2' U'

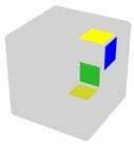
M' U2 M U' M2' U'

M' U2 M' U' M2' U'

U2 M' U2 M U M2 U'



U’ M’ U2 M U’

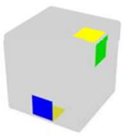


U M U2 M’ U

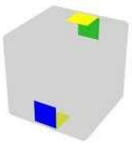
U M U2 M U

U’ M’ U2 M’ U’

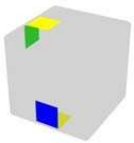
U2 M’ U2 M’ U’ M2’ U



U’ M U2 M U’



M’ U2 M’ U’ M2’ U



U M’ U2 M U

M’ U2 M U M2’ U

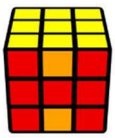
U M’ U2 M’ U

U2 M’ U2 M U’ M2’ U’

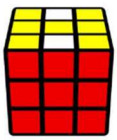
U’ M U2 M’ U’

Edge Permutation (EP)

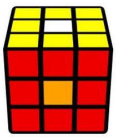
The last step is easy and intuitive. The small picture is the bottom layer viewed from the bottom.



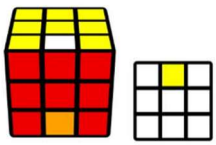
U2 M2’ U2 M2’



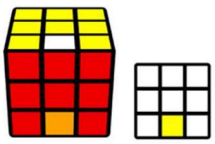
M’ U2 M2’ U2 M’



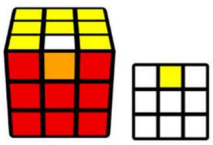
E2’ M E2’ M



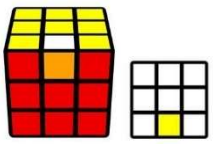
M’ U2 M U2



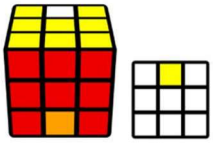
M2’ U2 M U2 M



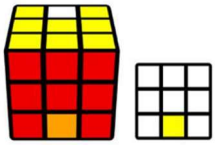
M’ U2 M’ U2 M2’



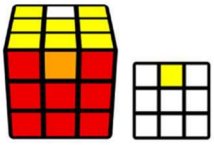
U2 M U2 M’



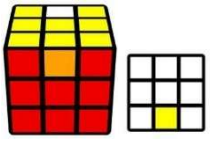
M2’ U2 M’ U2 M’



M U2 M U2 M2’



U2 M’ U2 M



3

Wanting to step into the field of advanced speedcubing? This chapter focuses on the advanced techniques in the CFOP system, including multi-angle F2L, non-standard F2L, lookahead, OLS, VH, and ZB. It also introduces other F2L solving structures, such as ZZ and Petrus. You can use this chapter as a reference to strengthen any part of your CFOP skills.

Prerequisite: Strong mastery of the Intermediate CFOP Method (preferably sub 12s average in 3×3×3 speedcubing)

3.1 First Two Layers (2)

The second part of the F2L tutorial is for intermediate cubers that have proficiently mastered CFOP algorithms and techniques covered in F2L (1). If you're aiming for further improvement, this tutorial is crucial for breaking the sub 10 barrier and stepping into the advanced level.

This tutorial covers:

- Good habits for solving the cross
- Solving standard F2L in multiple angles and in various ways
- Solving some non-standard F2L cases in different slots and angles
- F2L lookahead skills
- Last slot skills and OLL prediction

Cross

Here are three key criteria for mastering an expert-level cross:

1. Take 5~10 seconds to determine the best way to solve the cross.
2. Solve the cross in 0.5~1.5 seconds, and can be blindfolded.
3. Be proficient with both the white and yellow cross.

Don't start the timer right away if you find the cross easy. Instead, track the first F2L pair, so you can solve it seamlessly after completing the cross. You can use one or two additional moves to influence the F2L pair you're tracking (especially when you find a free pair opportunity).

Sometimes, you can combine these two steps, a technique known as **XCross**. Two common methods for creating an XCross are building a 1×2×2 block or using the Pseudo Slotting technique. However, don't aim for this if the cross is complex or involves multiple generators that make it hard to track.

You can also learn by studying how Rubik's Cube experts solve the cross, paying special attention to how they control the first F2L pair. Here are some former WR examples (they are somewhat old considering the speed people are breaking the records, but the solutions are very valuable for intermediate players to learn).

Max Park's Single WR (Jun 2023): 31 STM / 3.13s = 9.9 TPS

Scramble (Top white, front green): D R' U2 F2 D U' B2 R2 L' F U' B2 U2 F L F' D'

x2 R' D2 R' (BO corner PScross) **D L' U L** (Insert BL edge and GR corner simultaneously)
D R' U' R D (Remove the BO corner from BR slot and insert GR edge) //XXcross
L U' L' //OG 3rd pair **U' R U R' d R' U' R** //RB 4th pair
r' U' R U' R' U2 r U //OLL PLL Skip

Analysis: This solution is an excellent example of how Pseudo Slotting can be used to achieve XCross. Max Park derived the entire XXCross during the inspection time. The third free pair and a PLL skip also contributed significantly to this incredible world record.

Geng Xuanyi's Single WR (Apr 2025): 33 STM / 3.05s = 10.8 TPS

Scramble: R2 D2 R2 D2 U' F2 D L2 B2 R' D R B2 U2 L R2 D U2 F R2

R2' F L' (control RB free pair) **U' R** //Cross **U** (control BO free pair)
R' U2' R //RB **U L U' L** //BO **U L' U' L** //OG **R U2' R' U' R U R'** //GR
F R' F' r U R U' r' U //ZBLL-T

Analysis: This solution demonstrates the skill of controlling free pair in both Cross and F2L.

Wang Yiheng's Average WR (4.09s, Sep 2024)

#1 (46 STM / 3.83s = 12 TPS) Scramble: B R2 U2 B2 L' F2 R B' L R2 D' L U' R' D' R2 B2 R2

x2 y D L F' D' //Finish 3 edges and control 2 free pairs
U L F' L' F R U L U' L' //finish 2 free pairs and LCE **R' F R F' R U' R2' U R** //last 2 pairs
U2' r U R' U R U2' r' //OLL **R U' R' U' R U R D R' U' R D' R' U2 R'** //PLL

#2 (53 STM / 5.16s = 10.3 TPS) Scramble: B' D2 F2 D L' D2 U B D R' B2 D R2 F2 L2 U B' U'

y2 U' L' F' D' R' F R2 //Cross (Lookahead BO free pair)
U2 y L' U' L //BO **UD' R' U' R u** //RB **U L' U L R U' R'** //GR **U' L F' L' F U' L' U L** //OG
l' U' L U' L' U2 l //OLL **U' R2' U R U R' U' R3 U' R' U R' U'** //PLL

PLL skip opportunity missed: **U L' U L y' R U R'** //OG **F' r U R' U r' F R U2** //OLL PLL Skip

#3 (52 STM / 4.55s = 11.4 TPS) Scramble: U' L' B2 F' R' B2 F' R2 F U' F2 U' R F D L U2 R2

D L' D' L R' F' R //Cross and control BO free pair
d L' U2 L //BO **U' R U R2' U' R** //GR **U R U' R' y' L' U' L** //RB **R' U' R U R' U2' R** //OG
U r' U2' R U R' U r //OLL **U' L' U' L F L' U' L U L F' L2' U L** //PLL

#4 (36 STM / 3.71s = 9.7 TPS) Scramble: B' L2 R B F U R B' D2 B2 L' R U' L2 F L' D2 U2

x z2 F' U'D r' R F' //RB edge PScross **U R U R' D2** //RB XCross
y U2' R U R' U' L U L2' U L //GR & OG **U R U R' U R U' R'** //BO
R U2 R' U' R U' R' //OLL PLL Skip

#5 (61 STM / 3.88s = 15.7 TPS) Scramble: L' U' F' L U' L' U' R' F D2 L2 R D2 U L B F' R2

z2 y R' U' R2 (control GR free pair) **F' D U L U' L'** //F3E+GR **U' L F' L'** //XCross
R U' R' U R' U2' R //RB **F U F' U' L' U L** //OG **U R U' R' U' R U R'** //BO
U' R' F R U R' F' R F U' F' //OLL **R U R' U' R' F R2 U' R' U' R U R' F' U** //PLL

Analysis: The last four solves are very good examples since they involve both the block method and PSCross. You can also focus on the flexibility of Yiheng's solutions for different F2L cases.

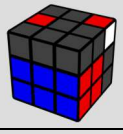
Multi-angle F2L

To minimize unnecessary whole-cube rotations, it's crucial to master how to solve F2L cases from multiple orientations and in different methods for convenience. Here's a comprehensive table that demonstrates multi-angle solutions for all 41 standard F2L cases. We will follow the same sequence and numbers as the one in F2L (1). Useful and important ones are highlighted. Some algorithms involving moves outside the R, U or L, U generators are optional, as they're usually less suited for efficient hand tricks. Still, I recommend learning some of these algorithms for your dominant hand positions. Here are some prior reminders:

- Don't rotate the cube automatically after finishing a slot until you're sure it's necessary.
- Be flexible in using the Keyhole technique to create a free pair, which can reduce the number of moves.
- There may be multiple options for a given orientation and situation. Some may utilize another slot, while others will only affect one slot.

Note: Blue: Supplemental but useful algorithms; Dark red: Only applicable to your dominant hand; Bolded: basic algorithms; Bolded and blue: Recommended; Normal: supplemental; Italic: involving other slots, excluding Keyhole technique.

#	1. FR Slot	2. FL Slot	3. BR Slot	4. BL Slot
1a				
	U2 R U R' (U R U' R') ***L U2 L' R U2' R'*** *(BL)*	**F R U2 R' F'** U2 L' U' L F U F'	y or y'	**U2 L U L' U L U' L'** ***R U2' R' L U2 L'*** *(FR)*
1b				
	F' L' U2' L F' U2 R U R' F' U' F'	**U2 L' U' L (U' L' U L)** ***R' U' R L' U2 L*** *(BR)*	**U2 R' U' R U' R' U R** ***L' U L R' U2' R*** *(FL)*	y or y'
2a				
	U R U2' R' (U R U' R')	y or y'	y or y'	**U L U2' L' U L U' L'**
2b				
	y or y'	**U' L' U2 L (U' L' U L)**	**U' R' U2 R U' R' U R**	y or y'

3a				
	R U2 R' U' R U R'	y or y'	y or y'	**L U2 L' U' L U L'**
3b				
	y or y'	**L' U2' L U L' U' L**	**R' U2' R U R' U' R**	y or y'
4a				
	U R U' R' U' → 17a1 *U2 (L F' L' F) R U R'* *(FL)*	**F' (U' L' U L) F L' U L** **(F R U R' F') d R U R'**	**R' U' R d'** → 17a1 y or y'	**L U L' U2'** → 18b4 **U L U' L' U'** → 17b4 U2 L2 U2 L' U' L U' L2'
4b				
	F (U R U' R') F' R U' R' **(F' L' U' L F) d' L' U' L**	**U' L' U L U** → 17b2 *U2 (R' F R F') L' U' L* *(FR)*	**R' U' R U2** → 18b3 **U' R' U R U** → 17b3 U2' R2' U2' R U R' U R2	**L U L' d** → 17b2 y or y'
5a				
	U R U' R' **R' F R F'**	**F' L F L'**	y or y'	**U L U' L'**
5b				
	F R' F' R	**U' L' U L** **L F' L' F**	**U' R' U R**	y or y'
6a				
	U' (R U R' U') R U2 R' **U' R U R' U2' R U' R'**	U l' U L U' L' U' l *L U L' (F' L F L') (BL)*	U r' U R U' R' U' r	**U' L U L' U' L U2 L'** **U' L U L' U2 L U' L'**

6b				
	r' U2' R2 U R2' U r *R' U' R (F R' F' R)* *(BR)*	**U (L' U' L U) L' U2' L** **U L' U' L U2 L' U L**	**U R' U' R U R' U2' R** **U R' U' R U2' R' U R**	U' l U' L' U L U l'
7a				
	U' R U2' R' U2 R U' R'	l U2 L2' U' L2 U' l' *L U2 L' (F' L F L')* *(BL)*	r U2' R2' U' R2 U' r'	**U' L U2' L' U2 L U' L'**
7b				
	U' r U' R' U R U r' *R' U2' R (F R' F' R)* *(BR)*	**U L' U2 L U2' L' U L**	**U R' U2 R U2' R' U R**	l' U2 L2 U L2' U l
8a				
	R U R' U2' → 17a1 R' D' R U' R' D R U R U' R' ***U' R' U R U' R U R'*** *(BR)* ***U L' U L U R U R'*** *(FL)*	**L' U L U2' (F U F')** *U2' R' U R (F U F')* *(BR)*	**R' U R U2' y R U R'** **R' U R U2' y' L U L'** ***U2 L' U L y/y'*** *(FL)*	**L U L' U2** → 17b4 ***U' L' U L U' L U L'*** *(FL)* *U R' U R U L U L'* *(BR)*
8b				
	R U' R' U2 (F' U' F) *U2 L U' L' (F' U' F)* *(BL)*	**L' U' L U2** → 17b2 L D L' U L D' L U' L U L ***U L U' L' U L' U' L*** *(BL)* ***U' R U' R' U' L' U' L*** *(FR)*	**R' U' R U2'** → 17b3 ***U R U' R' U R' U' R*** *(FR)* *U' L U' L' U' R' U' R* (BL)	**L U' L' U2 y' L' U' L** **L U' L' U2 y R' U' R** ***U2 R U' R' y/y'*** *(FR)*
9a				
	R U R'	**F U F'**	y or y'	**L U L'**

9b				
	F' U' F	**L' U' L**	**R' U' R**	y or y'
10a				
	U' R U R' U R U R' **U2 R U' R' U' R U R'** *R' U R2 U R' (BR)*	U L' U L U' (F U F') *L U L' (F U F') (BL)* y or y'	**U R' U R d' R U R'** *R U R' y/y' (FR)* y or y'	**U' L U L' U L U L'** **U2 L U' L' U' L U L'** *L' U L2 U L' (FL)*
10b				
	U' R U' R' U (F' U' F) *R' U' R (F' U' F)* *(BR)*	**U L' U' L U' L' U' L** **U2' L' U L U L' U' L** ***L U L2' U' L*** *(BL)*	**U R' U' R U' R' U' R** **U2' R' U R U R' U' R** ***R U' R2' U' R*** *(FR)*	**U' L U' L' d L' U' L** *L' U' L y/y' (FL)*
11a				
	U' R U' R' U R U R'	y or y' *L U' L' (F U F') (BL)*	y or y' *R U' R' y/y' (FR)*	**U' L U' L' U L U L'**
11b				
	y or y' *R' U R (F' U' F) (BR)*	**U L' U L U' L' U' L**	**U R' U R U' R' U' R**	y or y' *L' U L y/y' (FL)*
12a				
	R' U2' R2 U R2' U R **R U' R' U** → 15a1 *R' U2' R2 U R' (BR)*	**U L' U2 L U' (F U F')**	**U R' U2' R d' R U R'**	**L' U2 L2 U L2' U L** *L' U2 L2 U L' (FL)*
12b				
	U' R U2' R' U (F' U' F)	**L U2 L2' U' L2 U' L'** **L' U L U'** → 15b2 *L U2 L2' U' L (BL)*	**R U2' R2' U' R2 U' R'** *R U2' R2' U' R (FR)*	**U' L U2 L d L' U' L**

13				
	(U R U' R')3	(U' L' U L)3	(U' R' U R)3	(U L U' L')3
14				
	U' (R' F R F') R U' R'	U (L F' L' F) L' U L	R' U R d' R U' R'	L U' L' d L' U L
15a				
	U' R U' R' U2 R U' R'	U L' U2 L U' L' U' L	U' R' U' R U2 R' U' R	U' L U' L' U2 L U' L'
15b				
	U' R U2' U R U R'	U L' U L U2' L' U L	U R' U R U2' R' U R	U L U L' U2' L U L'
16a				
	U' R U R' d R' U' R	U' (F U F') U L' U' L	y or y'	U' L U L' d L' U' L
16b				
	U (F' U' F) U' R U R'	U L' U' L d' L U L'	U' R' U' R d' R U R'	y or y'
17a				
	R U' R' (U R U' R')	y or y'	y or y'	L U' L' U L U' L'
17b				
	y or y'	L' U L (U' L' U L)	R' U R U' R' U R	y or y'
18a				
	(R U R' U') R U R'	(L F' L' F) (U' L' U L) U2 L' U L (F U F')	y or y'	L U L' U' L U L'

18b				
	(R' F R F') (U R U' R') U2 R U' R' (F' U' F)	(L' U' L U) L' U' L	R' U' R U R' U' R	y or y'
19a				
	U' (R' F R F') R U R'	(U' L' U L) (F' L F L') (U' L' U L) d R U' R'	U' R' U R y (U R U' R')	L' U' L' U' L' U L U L
19b				
	(U R U' R') (F R' F' R) (U R U' R') d' L' U L	U (L F' L' F) L' U' L	R U R U R U' R' U' R'	U L U' L' y' (U' L' U L)
20a				
	R U' R' → 6a1	L' U L → 2b2	R' U R → 2b3	L U L' U' → 3a4
20b				
	R U' R' → 2a1	L' U L → 6b2	R' U' R U → 3b3	L U' L' → 2a4
21a				
	R U' R' U → 10b2/3	L' U L → 1a1/2/4	R' U R → 1a1/4	L U' L' d → 10b2
21b				
	R U' R' → 1b1/2/3	L' U L → 10a1/4	R' U R d' → 10a1	L U' L' → 1b2/3
22				
	R' F R F' → 12a1 R U' R' → 7a2/3	L F' L' F → 12b2 L' U L → 7b1/4	R' U R d' → 7b1	L U' L' d → 7a2

Non-Standard F2L

We learned how to transform basic NSF2L cases into a free pair. However, sometimes excessive cube rotations are required to solve them. Here are some useful algorithms for specific cases. These algorithms provide various methods for pairing non-standard F2L cases. Red-highlighted algorithms are meant only for use with your dominant hand.

Edge in Wrong Slot, Corner in U

	R U' R2' U R y2 L U' L2' U L	R' F U' F' R y2 F R' U R F'	
	U (R' F R F') (R' U' R) y U' (L F' L2 U L U2') F	U (R' F R2 U' R' U2) F' y U' (L F' L' F) (L U L')	
	(U R U' R')2 (L U2 L') y (U' L' U L)2 (R' U2' R)	y' (U' R' U R)2 L' U2 L y2 (U L U' L')2 R U2' R'	
	R U R2' U' R y2 L U L2' U' L	R' U' R2 U R' y2 L' U' L2 U L	
	U2 R' U' R2 U R2' U' R y2 U2 L' U' L2 U L2' U' L y U' (L' U' L) (R U' R') y' U' (R' U' R) (L U' L')	U2 R U R2' U' R2 U R' y2 U2 L U L2' U' L2 U L' y U (R U R') (L' U L) y' U (L U L') (R' U R)	
	(R U' R') d (L' U' L) y2 (L' U' L) d (R U' R') y L' (F' U' F) L	(R U2' R') y (R U R') y2 L U2 L' y' R U R' y U2 L' U L y' R' U2' R	
	U (R U R') (L U L') y U (L' U L) (R' U R) y' U (R' U R) (L' U L)	U' (R U' R') (L U' L') y U' (L' U' L) (R' U' R) y2 U' (L U' L) (R U' R')	
	U2 (R' F R F') (L U L') U2 F' (L U L') F y' U (R' U' R) d (R U R') y U (L' U' L) d (L U L')	U (R U R') d' (R' U' R) y U2 (L F' L' F) (R' U' R) y U2 F (R' U' R) F' y2 U' (L U L') d' (L' U' L)	

Corner in Wrong slot, Edge in U

	U R U R2' U2' R **y2 U L U L2' U2 L**	**U' R' U' R2 U2' R'** **y2 U' L' U' L2 U2 L'**	
	U' (R U' R' U') (R' U' R) **y (L' U' L) (F' U' F)** **y2 U' (L U' L' U') (L' U' L)**	**U (R' U R U) (R U R')** **y (R U R') (F U F')** **y2 U (L' U L U) (L U L')**	
	(R U2' R' U) (R' U' R) **U2 (R U' R' U) (R' U2' R)** **y2 (L U2 L' U) (L' U' L)** **y2 U2 (L U' L' U) (L' U2 L)**	**(R' U2' R U') (R U R')** **U2 (R' U R U') (R U2' R')** **y2 (L' U2 L U') (L U L')** **y2 U2 (L' U L U') (L U2 L')**	
	L' (R U' R') L **y2 R' (L U' L') R** Mirror case omitted	**(R U2' R') y (R' U' R)** **y2 U2 (L U' L') y (L' U2 L)** Mirror case omitted	

Use the basic insert algorithms from the first part for other opposite slot cases. These can be applied from multiple angles. For example, **R U R' = y2 L U L'**.

Other Useful Algorithms

	R2 U' R2' U R2 **y2 L2' U' L2 U L2'**	**R2 U R2' U' R2** **y2 L2' U L2 U' L2'**	
	(R' F R F') (L' U L)	**R (L U2' L) R'**	
	R U' R2' U2' R	**R' F R U' F'**	
	R2 U2' R2' U2' R2	**F (r U r' U')3 F'**	

F2L Lookahead

Two major factors affect solving time: move count and TPS (turns per second). Intermediate solvers often use about 50–70 moves, while advanced solvers may use a similar number of moves but solve much faster because of higher TPS, smoother turning, and fewer pauses.

Although OLL and PLL usually require short pauses for recognition, F2L gives the most room for improvement. **Lookahead** means finding or predicting the next F2L pair while solving the current one. It is not just multi-slotting; it is the ability to anticipate the next pair, recognize whether it is standard or non-standard, and understand the orientation of its pieces before you reach it.

The following table shows the stages of lookahead skills:

Stage (level)	Algorithms/Skills	Description	Suggestions
1. Entering Intermediate (Avg. 15-20s)	Standard F2L, with some Back F2L (all PLL, some OLL)	Recognition is still not proficient; lookahead is limited	Focus on improving case recognition and lookahead awareness
2. Middle Stage (Avg. 12-15s)	All Standard F2L Some MA & NSF2L (all OLL & PLL)	Can often predict the next pair during both Cross and F2L; connects easy cases smoothly	Focus on MAF2L and NSF2L and how to apply them; learn more skills
3. Proficient Stage (Avg. 10-12s)	All Standard F2L Most MA & NSF2L Some last slot skills	Maintains fluency at a medium to fast pace; connects standard cases smoothly, handles many NSF2L cases	Focus on practicing MAF2L and NSF2L, reducing moves, and learning some advanced skills
4. Entering Advanced (Avg. 8~10s)	All Standard F2L Common NSF2L Some XCross and multi-slotting skills	Maintains fluency at a fast pace, connects quickly in most cases, and achieves seamless transitions for easy cases	Focus on intensive practice and advanced techniques such as MW, WV, SV; optional to start ZB learning
5. Advanced level: Needs highly consistent lookahead, efficient solutions, strong turning control. Advanced last-slot and last-layer techniques are often required.			

Lookahead Strategies

When choosing the next pair, you can use this priority order:

1. Look for an existing pair in the U layer before finishing the current pair.
2. Look for a pair where one piece is in the U layer and the other is in the current slot.
3. If a piece is already in its correct slot, look for its matching piece in the U layer or current slot. These cases are often easy to connect.
4. If none of the above works, the next pair may be non-standard. Track a piece in the U layer or current slot, preferably a corner, and find its matching piece in an incorrect slot.
5. For non-standard or multi-slot cases, remember that a U-layer piece may swap into a slot, and a slot piece may move into the U layer.

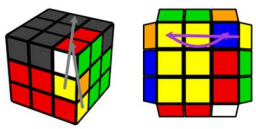
Practicing Lookahead

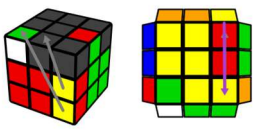
To improve lookahead, slow your turning slightly so you have time to track the next pair while solving the current one (useful even in timed solves). Practice without a timer at first, then gradually increase speed while keeping consistency.

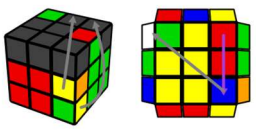
Studying strong solvers can also help. Pay attention to how they solve the cross and F2L, how they reduce pauses, and how they transition smoothly from one pair to the next. Observing their approach can help you refine your own skills in recognition and anticipation.

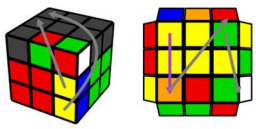
Prediction Arrows

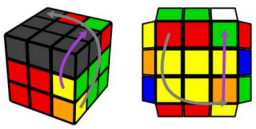
The arrows in the cases below show where the pieces go after standard F2L operations. You don't need to memorize them, but it is helpful to pay attention to the movement of pieces during practice.

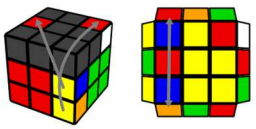

U R U' R'

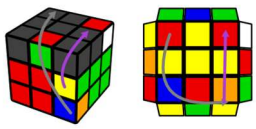

U' R U R'

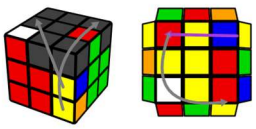

(R U' R' U') (R U R')

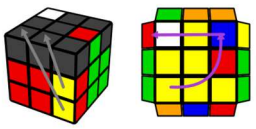

R' U2' R2 U R2' U R

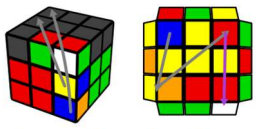

(R U' R' U) (R U R')

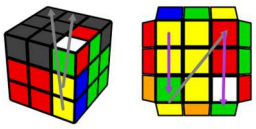

(R U R' U2) (R U' R')

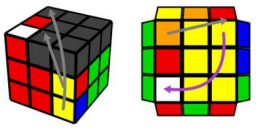

(R U2' R' U2) (R U' R')

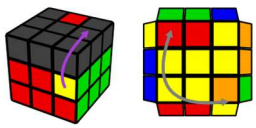

(R U2' R' U) (R U' R')

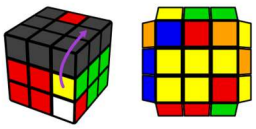

(R U R' U) (R U' R')

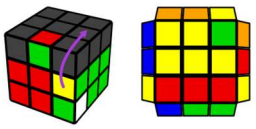

(R U R' U2') (R U' R' U)
(R U' R')

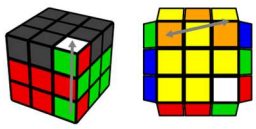

(R U2' R' U') (R U R')

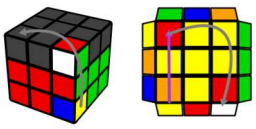

(R U' R' U') (R U' R' U)
(R U' R')

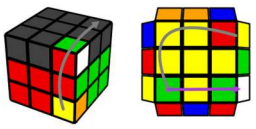

F' (R U R' U') (R' F R)


U (R U' R' U) (R U' R')


U' (R U R' U') (R U R')


(U R U' R')3


U' (R U' R' U2) (R U' R')


U (R U R' U2') (R U R')

3.2 Last Slot Techniques

When solving the last F2L pair, consider the connection to OLL to reduce recognition time and, in some cases, control OLL for an easier solution.

Basic Skills

Keep the last slot in the front if it can be done naturally
This would simplify recognition of the last pair as well as allow for the use of VHLS, MW, or WV algorithms, which are only applicable when the last slot faces you.

Predict the Orientation of the Last Layer Edges
When only two generators (RU or LU) are used to solve the last pair and the edge is in U, observe the current yellow edges (if yellow is the top layer):

- 3 edges oriented → 4 edges will be oriented after solving the last slot.
- 1 or 2 edges oriented → 2 edges will be oriented after solving the last slot.
- 0 edges oriented → 0 edges will be oriented after solving the last slot. In this case, switch to the third technique.

If the edge of the last slot is correctly placed, the orientation of the yellow edges does not change. Note that this prediction method doesn't work if 3 generators is used to solve the last pair.

Avoid Dot OLL
Dot OLL cases require the most complex algorithms, whereas OCLL algorithms (edges are oriented) are simpler. If a connected free pair remains with no oriented yellow edges, apply the algorithms below to avoid a dot OLL or create an OCLL in certain situations. More techniques are introduced in the EOLS section.

R' F R F'
y F' L F L'

F R' F' R
y L F' L' F

R' F R F'
y F' L F L'

F R' F' R
y L F' L' F

OLS Techniques

OLS are advanced algorithms that solve the last slot and OLL simultaneously. It includes VLS (Valk Last Slot, when the last pair is a connected free pair), CLS, BLE, FLS, OOLS, and 123OLS.

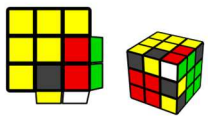
This section introduces three common sets of last slot algorithms: MW (Magic Wonderful, VLS subset with a 2×2 block in the yellow face), WV (Winter Variation, VLS subset with all edges oriented), and SV (Summer Variation, separated free pair with all edges oriented). Most of these algorithms are useful and can increase efficiency significantly.

A sub 10 speedcubing level is strongly recommended for learning OLS techniques.

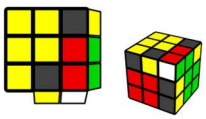
Note: Blue algorithms are recommended; green algorithms are OCLL variants (labeled specifically); the yellow part of an algorithm is OLL. MW and WV algorithms include both right-hand cases and left-hand cases, but they are designed for right-handed cubers.

MW Algorithms (18)

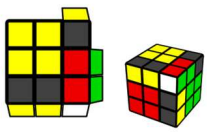
Pure MW



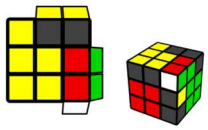
U (F' U' F U) R U2 R'



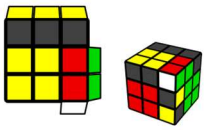
M' U R U' r'



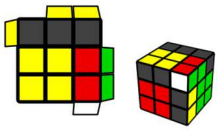
R' F R F'



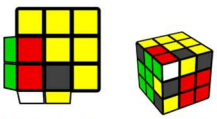
U (F' U F) R U' R'



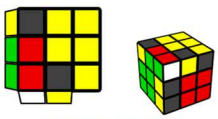
U F' (L' U2 L U) F



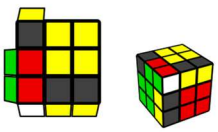
U R B U' B' R'



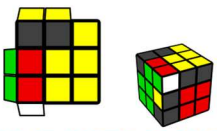
U' (F U F' U') L' U2 L



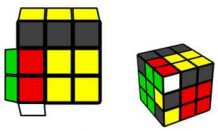
M' U' L' U l



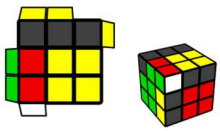
L F' L' F



U' (F U' F') L' U L

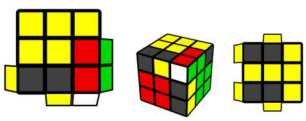


U' F (R U2 R' U') F'

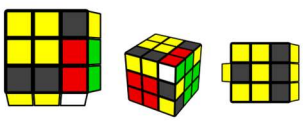


U L' B' U B L

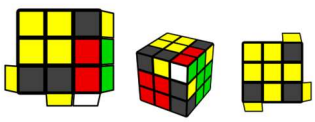
MV Special Insert



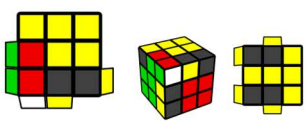
(U R U' R')
(F R U R' U' F')



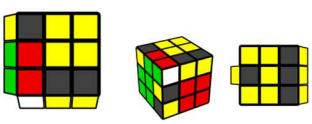
U R U' R' U'
(R' U' R' F R F' U R)



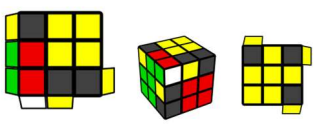
(R' F R F')
(R' U' R U' R' U2 R)



(U' L' U L) U2'
(F R U R' U' F')



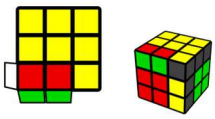
U' L' U L U
(R' U' R' F R F' U R)



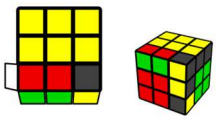
(L F' L' F) U2
(R' U' R U' R' U2 R)

Right Hand WV Algorithms (27)

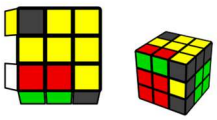
3/2 Corners Oriented



L' U2 (R U R')
U2' L

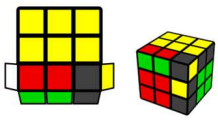


R U' R'

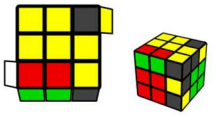


S

(R U R' U')
(R U' R')

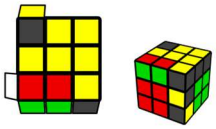


U' (R' F R U)
(R U' R' F')

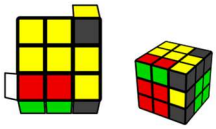


U

R2 D (R' U' R D') R2'

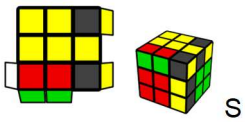


U' (R U' R' U2)2 (R U R')



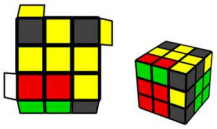
(R U' R') U (R' U' R U' R' U2 R)

WVR: 1 Corner Oriented



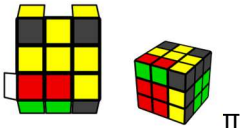
S

U R U2' R'



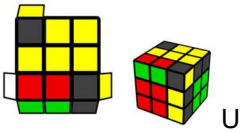
H

U (R U' R' U)
(R U2' R')



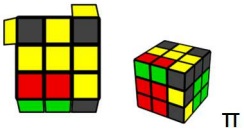
π

R U (R2' U' R2 U')
(R2' U2' R)



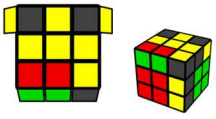
U

U R2 D
(R' U2 R D') R2'

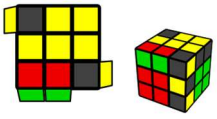


π

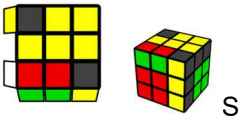
U R' U' R2 U'
R2' U2' R



U F' (R U2' R' U2')
(R' F R)

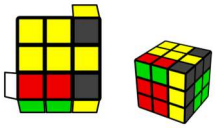


U' L' (U R U' R') L

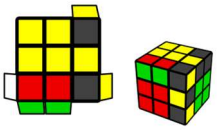


S

U R U2' (R2' U' R
U' R' U2 R)

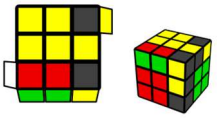


(R U' R' U2)
(r U R' U' r' F R F')

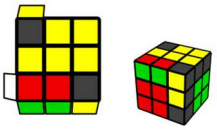


U

(R U' R)
(D' R U2
R' D R U2 R)

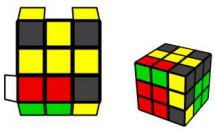


(R U' R') U (R2 D R'
U2 R D' R' U2 R')

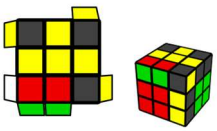


(R U' R' U)
(r U R' U' r' F R F')

No Corner Oriented

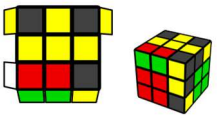


U (R U' R' U)2
(R U2' R')



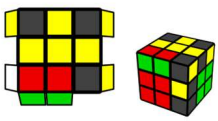
H

(R U R' U')2 R U' R'

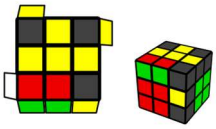


U

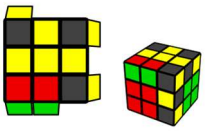
R2 D (R' U R D')
(R' U2 R')



(R U' R') U2
(R U2' R2' U' R2 U'
R2' U2' R)

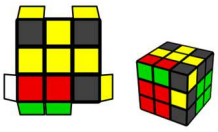


(R U' R') U'
(R U R' U R U2' R')



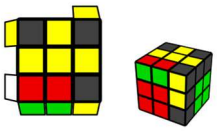
S

R U' (R2' U2' R U
R' U R)



S

U R U2' (R2' U2' R
U R' U R)



S

R U' (R2' U' R U' R'
U2 R)

Left Hand WV Algorithms (27)

3/2 Corners Oriented

R U2' (L' U' L)
U2 R'

L' U L

S

(L' U' L U)
(L' U L)

U (L F' L' U')
(L' U L F)

U

L2' D' (L U L' D) L2

U (L' U L U2)2 L' U' L

L' U L (R' U2' R U R' U R)

1 Corner Oriented

S

U' L' U2 L

H

U' (L' U L U')
(L' U2 L)

π

L' U' (L2 U L2' U)
(L2 U2 L')

U

U' L2' D'
(L U2 L' D) L2

U' L U L2' U
L2 U2 L'

U' F (L' U2 L U2)
L F' L'

U R U' L' U L R'

S

U' L' U2 (L2 U L'
U L U2' L')

L' U L
(r U R' U' r' F R F')

U

(L' U L) (R2' D' R
U2 R' D R U2 R)

(L' U L U') (R2 D R'
U2 R D' R' U2 R')

(L' U L U)
(r U R' U' r' F R F')

No Corner Oriented

U' (L' U L U')2 (L' U2 L)

H

(L' U' L U)2 L' U L

U

L2' D' (L U' L' D) (L U2 L)

(L' U L) (R U2' R2' U' R2 U' R2' U2' R)

L' U L (R U2' R' U' R U' R')

S

L' U (L2 U2 L' U' L U' L')

S

U' L' U2 (L2 U2 L' U' L U' L')

S

L' U (L2 U L' U L U2 L')

SV Algorithms (27)

Although Summer Variation is often considered less useful than Winter Variation, it is still helpful for advanced cubers that pursue fluency between F2L and OLL.

Note: Purple algorithms are WV algorithms.

3/2 Corners Oriented

U

(R U R D) (R' U2 R D') (R' U2 R')

(R U R' U) (R' U2' R U R' U R)

(R U' R' U') R2 D (R' U' R D') R2

U2 L (R U' R' U) L'

R U R'

T

(R U R2' F') (r U R U') r' F

(R U R' U') (R U R' U R U2' R')

1 Corner Oriented

(R U2' R' U') L' (U R U' R') L

(R U2' R' U') (R' F R U) (R U' R' F')

(R U' R' U') (R U' R')

(R U2' R' U') (R U' R' U2)2 (R U R')

(R U’ R’ U’) (R U R’ U’)2 (R U’ R’)

S

R U (R2’ U’ R U’ R’ U2 R)

U

(R U’ R D) (R’ U2 R D’) R2

U2 (L U L’ U’) (r U2 L U2) r’ L’

(R U’ R’ U’) (R U R’ U’) (R U’ R’)

(R U2’ R’ U) F’ (R U2’ R’ U2’) (R’ F R)

R U’ (R2’ U’ R2 U’ R2’ U2’ R)

L’ (R U R’ U’) L

No Corner Oriented

(R U’ R’ U’) (R U’ R2’ U’ R U’ R’ U2’ R)

U’ (R U R D) (R’ U R D’) R2

(R U2 R D) (R’ U’ R D’) R2’

S

R U (R2’ U2’ R U R’ U R)

U2 L (R U’ R2’ U) L’ U’ R

(R U R’ U’) (R U2’ R’ U’ R U’ R’)

(R U R’ U) (R’ U’ R U’ R’ U2’ R)

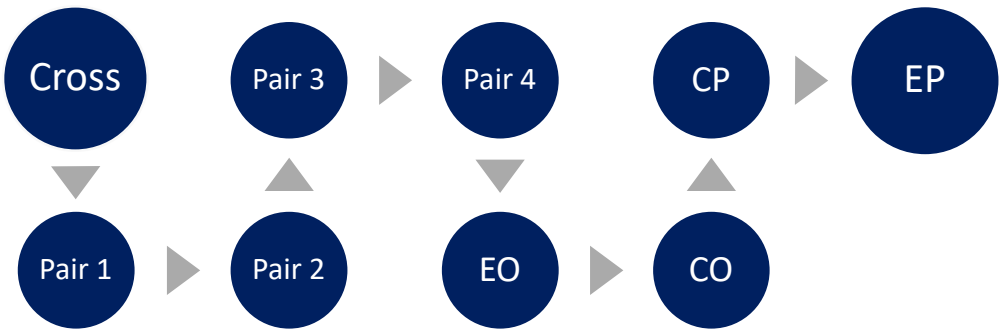
(R U2’ R’ U) (R U’ R’ U) (R U2’ R’)

***The core of optimizing method – Merging processes**

XCross = Cross + Pair 1
VHLS = Pair 4 + EO
COLL = CO + CP

OLL = EO + CO
WV = Pair 4 + CO
2GLL = CO + EP

PLL = CP + EP
MW = Pair 4 + OLL
ZBLL = CO + PLL



EOLS Techniques

EOLS algorithms are designed to support COLL (CO + CP, a subset of ZBLL), 2GLL (CO + EP, a subset of ZBLL), or ZBLL (CO + PLL), because they require EO to be solved. This solving structure is used by many strong solvers because the solution is much shorter than regular methods and the number of algorithms is manageable for them.

This section only covers VHLS, an EOLS subset that solves the last free pair and edge orientation simultaneously. VHLS algorithms are usually followed by COLL + EPLL, which the entire solving structure is called the **VH Method** (introduced in 3.3). For ZB-level solving, more EOLS algorithms are desired. Learning VHLS is also helpful for CFOP players as they can help avoid dot OLL cases and increase the probability of OCLL.

VHLS Algorithms (32)

VHLS cases are very easy to recognize by simply looking at the yellow edges on the U face. You can try to look ahead to VHLS while transforming the last pair into the free pair. As mentioned in basic techniques, it is important to leave the last slot in the front in order to perform a VHLS algorithm. If you are not allowed to do so, some algorithms can be done from other angles.

Note: Green algorithms are those recommended for right-handed CFOP cubers.

Connected Form

U R U' R'

U F' L' U' L F

U R B U' B' R'
y2 U L F U' F' L'

R' F R F'

U S' R U' R' S
y2 U S L U' L S'

R' F R2 U R' U' F'

U (R U R' U2')
R' F R F'

y (R' F' R U2)
M' U' M

U' L' U L

U' F R U R' F'

U' L' B' U B L
y2 U R' F' U F' R
y' U' F R' F' R2 U' R'

L F' L' F

U’ S L’ U L S’
y2 U’ S’ R’ U R S

L F’ L2’ U’ L U F

U’ (L’ U’ L U2) L F’ L’ F
y’ U R’ D’ r U’ r’ D R

y’ (r U r’ U2) M’ U M

Separated Form

R U R’

U2’ r U’ r’ U’ r U r’

S (R U R’ U’) S’
y2 S’ L U L’ U’ S

(R U R2’ U’)
(F’ U F R)

R U2’ R’
(F’ L’ U’ L F)

R U2’ B U’ B’ R’
y2 L U2’ F U’ F’ L’
y F U2’ R U’ R’ F’

S’ R U R’ S
y2 S L U L’ S’
y S’ (U’ L’ U L) S
y’ S (U’ R’ U R) S’

R’ D’ (r U r’ D)
R2 U R’

L’ U’ L

U2 (R’ F R U)
R’ F’ R

S’ (L’ U’ L U) S
y2 S R’ U’ R U S’

y’ U (R U2’ R’ U’)
F’ U F

L’ U2 L
(F R U R’ F’)

L’ U2 B’ U B L
y2 R’ U2’ F’ U F R
y F’ U2 L’ U L F

S L’ U’ L S’
y2 S’ R’ U’ R S
y’ S (U R U’ R’) S’
y S’ (U L U’ L’) S

y’ R D (r’ U r D’)
R2’ U’ R

3.3 Advanced Methods

Top solvers have continued to optimize beyond standard CFOP. Popular advanced speedsolving methods include ZB, ZZ, and Petrus. They use different approaches, but they share the same goal: orient the last-layer edges earlier, so the cube can be solved with a single **ZBLL algorithm** (493 cases in total). Alternatively, OCLL+PLL or COLL+EPLL (VH Method) can be used.

Among these methods, ZB is the most popular for advanced 3×3×3 speedcubing because its F2L structure is closest to CFOP. ZZ and Petrus are commonly used in FMC because of their efficient block-building and edge-orientation ideas. This tutorial will present **COLL** and **2GLL** algorithms as an exposure to ZB learning and introduce ZZ and Petrus briefly. It is beneficial for a CFOP solver to know some ZZ/Petrus ideas, especially if you'd like to try FMC.

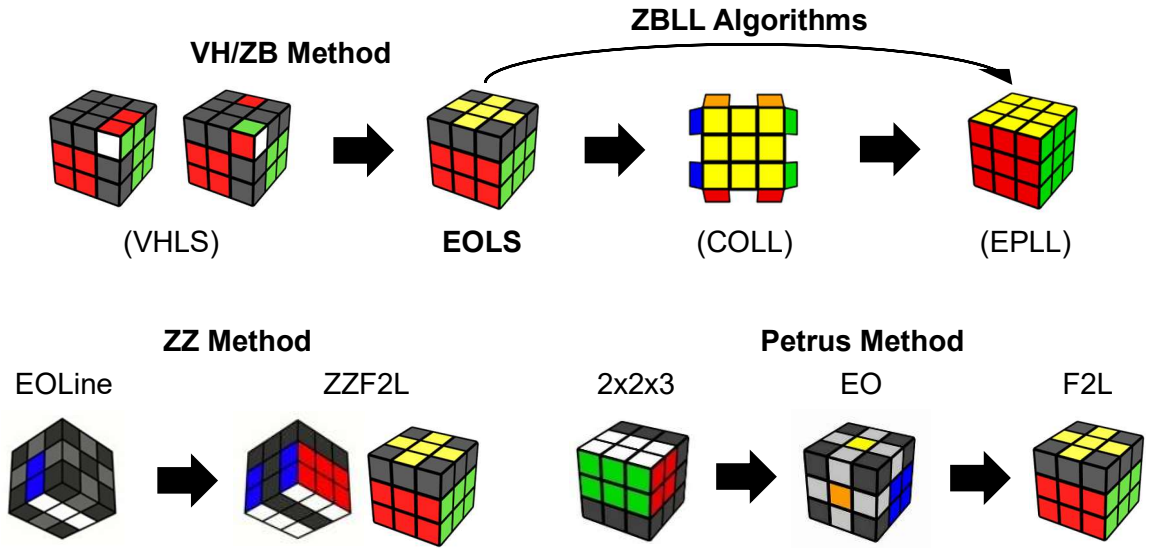
Comparing the Methods

VH Method is a simplified version of ZB. It uses VHLS and breaks ZBLL into COLL and EPLL. It is the foundation for learning ZB, but it takes more moves than regular CFOP, CFCE, or Roux.

ZB Method is the advanced CFOP-based method that is used by many top speedcubers. It follows traditional F2L solving, and orients the top edges while solving the last slot with EOLS/ZBLS algorithms. This method is structured, predictable, and compatible with a CFOP background.

ZZ Method solves the edge orientation at the very beginning by EOLine. After EO, the first two layers can be solved only with R, U, and L moves, usually without cube rotations. ZZ offers an ergonomic F2L and LL solving in exchange of a difficult EOLine step.

Petrus Method builds a 2×2×3 block first, fixes the unoriented edge in the middle, and then completes the F2L. Its main advantage is efficient block-building, but the two-sided F2L stage can feel less consistent and ergonomic for speedcubing.



VH Method

The **VH Method** (Vandenbergh-Harris) is considered a simplified version of the ZB Method, using VHLS and COLL. The VHLS algorithms are in 3.2. For a pure VH solver, you can apply either OCLL + PLL or COLL + EPLL (the normal VH solving structure) after finishing VHLS.

VH is generally the prerequisite for learning ZB since COLL is a subset of ZBLL algorithms.

COLL Algorithms (40)

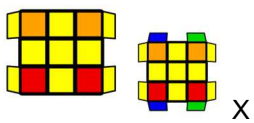
Recognition and Memorization: Similar to CLL, you should become very familiar with opposite and adjacent colors. COLL is slightly easier for recognition than CLL since the edges are ordered. S+ and S- cases are very hard to recognize, so using OCLL + PLL combination for those cases would be the better choice if you struggle with recognizing them. Several notes:

(1) Each case has 2 pictures: the larger one helps with recognition and the small one shows the detailed case. Follow the **orientation of the smaller picture**.

(2) The small letters beside the image indicate which two corners need to be swapped. X = Corners permuted. D = Diagonal. B/F/L/R = the back/front/left/right pair.

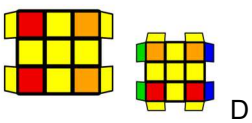
(3) Yellow: OLL variants. Blue: Overlapped with 2GLL (two-generator) algorithms.

H



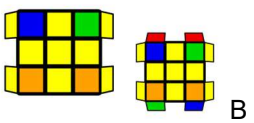
X

(R U R' U) (R U' R' U) R U2' R'



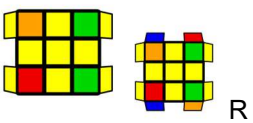
D

F (U R U' R')3 F'



B

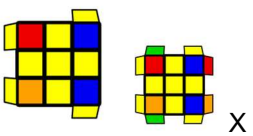
F (R U' R' U) (R U2 R' U') (R U R' U') F'



R

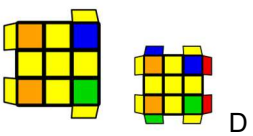
(R U R' U R U) (r' F R' F' r)

π



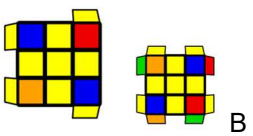
X

R U2' R2' U' R2 U' R2' U2' R



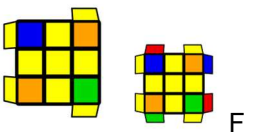
D

(r U R' U) (R' F R F') (R U' R' U) R U2' r'



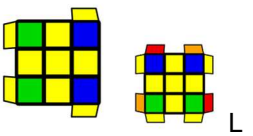
B

F U (R U' R' U) (R U' R2' F') (R U R U' R')



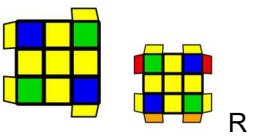
F

(R U R' U') (R' F R2 U R' U') (R U R' U') F'



L

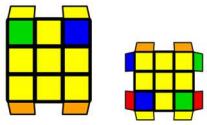
(R U R' U) F' (R U2' R' U2) R' F R



R

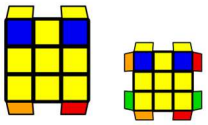
F U (R U' R' U) (R U2' R' U') (R U R' F')

U



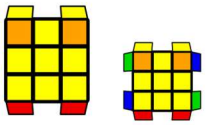
X

(R’ U’ R U’ R’ U2’)
(R2 U R’ U R U2’ R’)



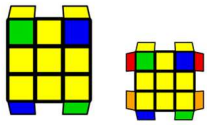
D

(R’ U2’ R F) U’
(R’ U’ R U) F’



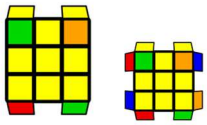
B

R’ (F R U’ R’ U’ R U R’ F’
R U R’ U’ R’ F R F’) R



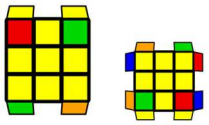
F

F (R U’ R’ U) (R U R’ U)
(R U’ R’ F’)



L

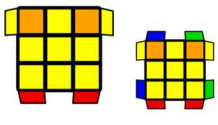
R2’ D’ (R U2 R’ D) R U2 R



R

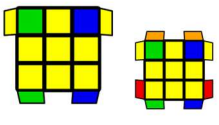
R2 D (R’ U2 R D’) R’ U2 R’

T



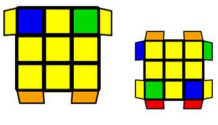
X

(R U2’ R’ U’ R U’)
(R2’ U2’ R U R’ U R)



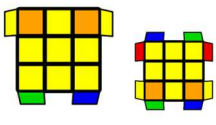
D

(R’ U R2 D) (r’ U2 r D’)
R2’ U’ R



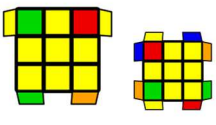
B

(R’ U R U2’) r’ (R’ F R F’) r



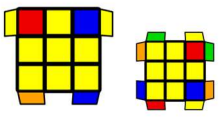
F

F (R U R’ U’) (R U’ R’ U’)
(R U R’ F’)



L

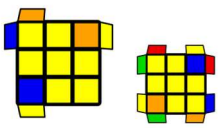
(r U R’ U’) (r’ F R F’)



R

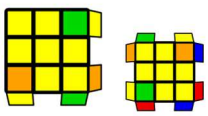
(R’ F’ r U) (R U’ r’ F)

L



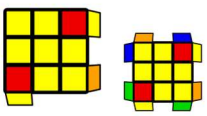
X

(R U R’ U) (R U’ R’ U)2
R U2’ R’



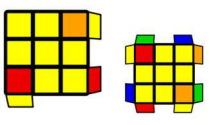
D

R’ U’ (R U R’ F’)
(R U R’ U’) R’ F R2



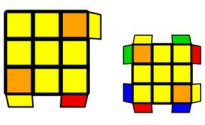
B

(R U2 R D)
(R’ U2 R D’) R2’



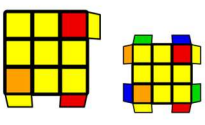
F

F’ (r U R’ U’) r’ F R



L

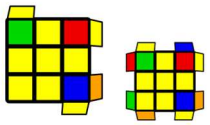
(R’ U2 R’ D’) (R U2 R’ D) R2



R

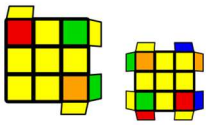
(F R’ F’ r) (U R U’ r’)

S+



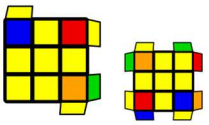
X

R U R' U R U2' R'



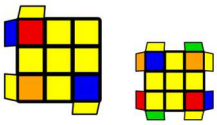
D

(R U R' U) L' (U R U') L U2' R'



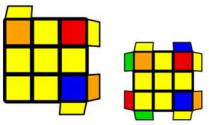
B

(R' U2' R U) F (R' U R U') F'



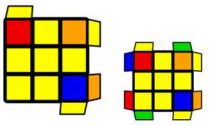
F

R2' D' (R U' R' D) (R U' R U) R' U R



L

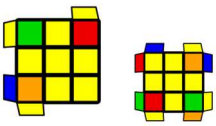
R U' L' U R' U' L



R

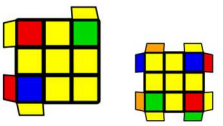
L' (R U R' U') L (U2 R U2' R')

S-



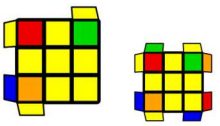
X

R' U' R U' R' U2 R



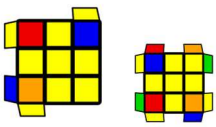
D

R U2' L' (U R' U') L (U' R U' R')



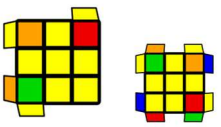
B

(R U2' R' U2) L' (U R U' R') L



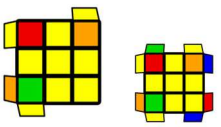
F

(R' U' R U') (R' U R' D') (R U R' D) R2



L

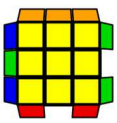
F (U R' U' R) F' (U' R' U2' R)



R

L' U R U' L U R'

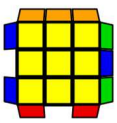
EPLL Algorithms (4)



1/18 Ua

R2' U (R U R' U') R3 U' R' U R')

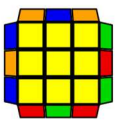
[U2] (R' U R' U') R3 U' (R' U R U R2'



1/18 Ub

M2' U M U2 M' U M2'

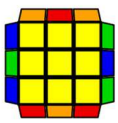
(R U R' U) (R' U' R2 U') (R' U R' U R)



1/36 Z

(M2' U)2 M' U2 M2' U2 M'

[U'] M' U (M2' U)2 M' U2 M2'



1/72 H

M2' U M2' U2 M2' U M2'

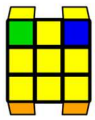
ZB Method

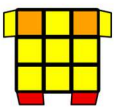
The **ZB Method** (Zborowski-Bruchem) is used by many top speedsolvers because the solution is very compatible with standard CFOP. Compared to VHLS (3.2), it uses deeper EOLS technique which can include hundreds of algorithms.

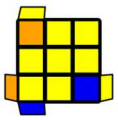
This section introduces **2GLL** algorithms as an exposure to the ZB Method (S+ and S- cases are excluded because they are hard to recognize). 2GLL are the last layer cases that can be solved using only two generators (R and U). In all 2GLL cases, the edges are already oriented and the corners are permuted, so we can interpret it as the set of algorithms that combines the process of CO and EP.

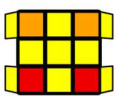
2GLL Algorithms (56)

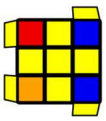
Recognition and memorization: You should first recognize the OCLL type (U, T, L, H, or π) and confirm that it is a 2GLL case, then distinguish the cases by bars, lights, or opposite colors.


2GLL - U


2GLL - T

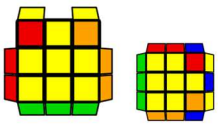

2GLL - L

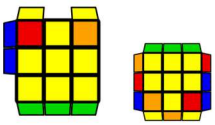

2GLL - H

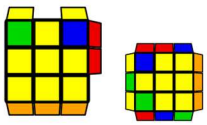

2GLL - π

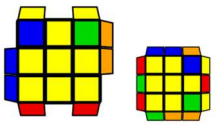
Note: (1) Refer to the larger picture for key observation points, the **smaller picture for orientation**.
(2) Blue: COLL algorithms; Yellow: Sune combinations.

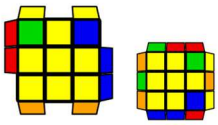
2GLL - U

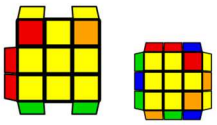

(R' U2' R U') (R U2' R' U2) (R' U' R U) R U' R'

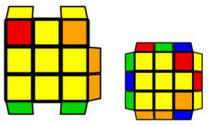

(R U R' U R U2' R') U (R U2' R' U' R U' R')

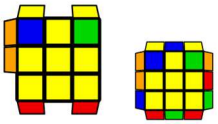

(R U R' U') (R U' R' U2)2 R U R'

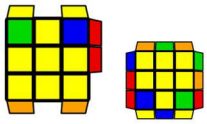

(R U2' R' U' R U' R') U' (R U R' U R U2' R')

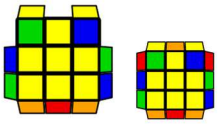

(R' U2 R U R' U R) U (R' U' R U' R' U2 R)

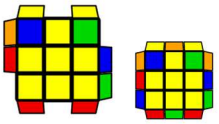

R' U2' R2 U R2' U R U' (R U R' U') R U' R'

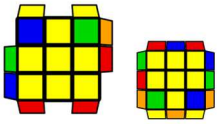

R U2' R2' U' R2 U' R' U (R' U' R U) R' U R


(R' U' R U' R' U2 R2) (U R' U R U2' R')

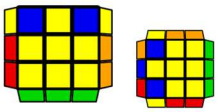

(R U R' U R U2' R2') (U' R U' R' U2' R)


(R U R' U') (R U' R U2') R2' U' (R U R' U') R2 U' R2'

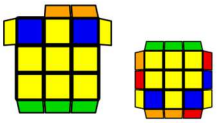

(R' U' R U') R U2' R2' U' R2 U' R2' U R


(R U R' U) R' U2' R2 U R2' U R2 U' R'

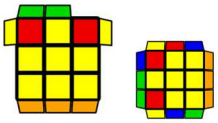
2GLL - T



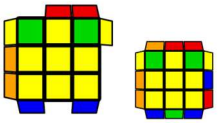
(R U R' U R U2' R') U2 (R' U' R U' R' U2 R)



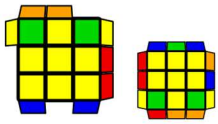
(R U' R' U2) (R U R' U2) (R U R' U) R U' R'



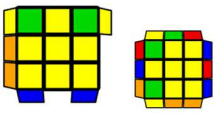
(R U R' U R U2' R') U' (R U2' R' U' R U' R')



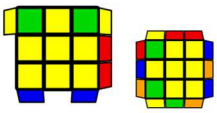
(R U2' R' U' R U' R') U (R U R' U R U2' R')



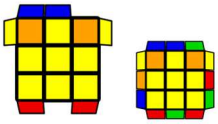
(R' U2' R U R' U R) U' (R' U' R U' R' U2 R)



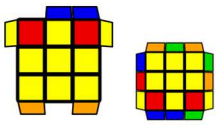
(R U R' U R U' R' U) R' U' R2 U' R2' U2' R



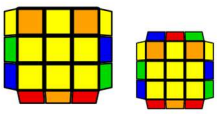
(R' U' R U') (R' U R U') R U R2' U R2 U2' R'



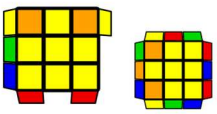
R U2' R' U' R U' R2' U2' R (U R' U R)



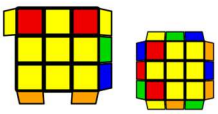
R' U2' R U R' U R2 U2' R' (U' R U' R')



(R' U R U2) (R' U' R U') (R U R' U') (R' U' R U) R U' R'

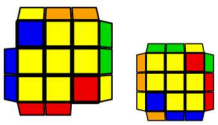


R' U' R2 U R2' U R2 U2' R' (U R' U R)

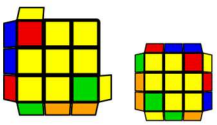


R U R2' U' R2 U' R2' U2' R (U' R U' R')

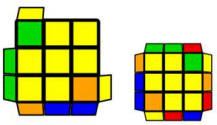
2GLL - L



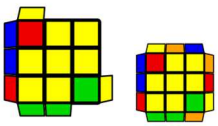
(R U2' R' U2) (R' U' R U) (R U' R' U2) R' U2' R



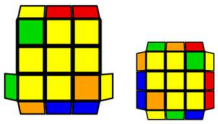
(R U R' U R U2' R') U2 (R U2' R' U' R U' R')



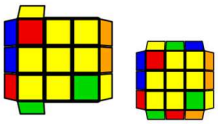
(R U2' R' U' R U' R') U (R' U2' R U R' U R)



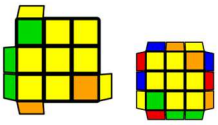
(R' U2' R U R' U R) U' (R U2' R' U' R U' R')



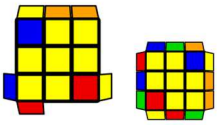
(R U R' U R U2' R') U (R' U' R U' R' U2 R)



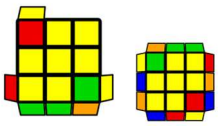
(R' U' R U' R' U2' R) U' (R U R' U R U2' R')



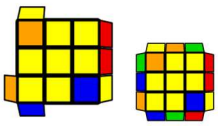
R2 U (R' U R' U') (R U' R' U') (R U R U') R2'



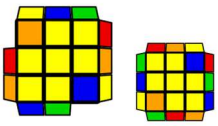
R2' U' (R U R U') (R' U' R U') (R' U R' U) R2



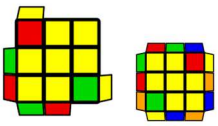
R2' U' (R U' R U R' U) (R U R' U') R' U R2



R2 U (R' U' R' U) (R U R' U) (R U' R U') R2

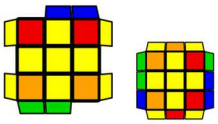


(R U2 R' U') (R U R' U')2 R U' R'

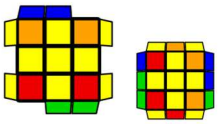


(R U2' R' U' R U' R') U2 (R U R' U R U2' R')

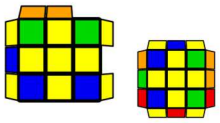
2GLL - H



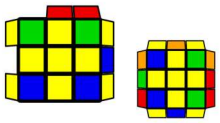
(R' U2' R U R' U R) U (R U R' U R U2 R')



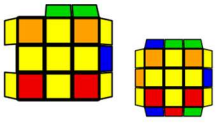
(R U2' R' U' R U' R') U' (R' U' R U' R' U2 R)



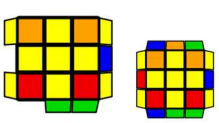
(R U2' R' U') (R U R' U') R U' R'



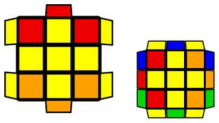
R' U2' (R U R' U') (R U R' U R)



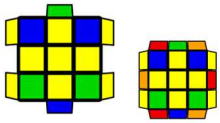
(R U R' U) (R U' R' U) R U2' R'



(R' U' R U') (R' U R U') R' U2 R

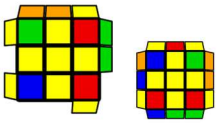


(R U R' U) (R' U' R U) R' U' R2 U' R2 U (R U2 R' U2' R)

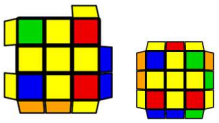


(R U R' U R U2' R') U' (R' U2' R U R' U R)

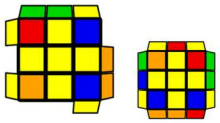
2GLL - π



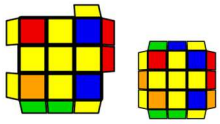
(R U R' U R U2' R2') (U2' R U R' U R)



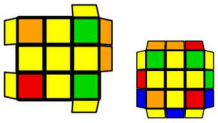
(R U2' R' U' R U' R2') (U' R U' R' U2' R)



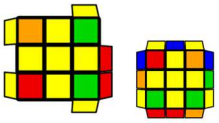
(R U2' R' U' R U' R') U (R U2' R' U' R U' R')



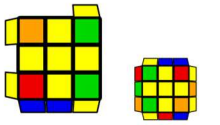
(R U R' U') R' U2' R U R' U R2 U2' R'



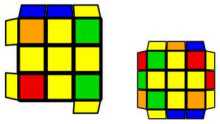
(R U2' R' U2) (R U' R' U2)2 R U R'



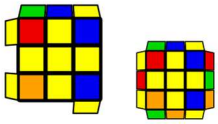
(R' U2 R U2') (R' U R U2')2 R' U' R



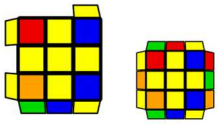
(R' U R U') R2' U2' (R U R' U) R2 U' R' U R



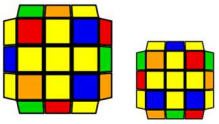
(R U' R' U) R2 U2' R' U' R U' R2' U R U' R'



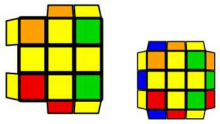
R U2' R2' U' R2 U' R2' U2' R



R' U2' R2 U R2' U R2 U2' R'



(R U R' U R U2' R') U (R U R' U R U2' R')



R U R2' U' R2 U' R2' U2' R2 U' R' U R U2' R'

ZZ Method

The **ZZ Method** was created by Zbigniew Zborowski in 2006. It was designed as an efficient and ergonomic method that combines block-building with layer-by-layer solving. Its defining feature is that the first step always includes Edge Orientation, or EO. ZZ is only used by a small portion of speedsolvers, but it is one of the most popular methods for FMC (fewest move) solving.

ZZ's main strength is that it makes the rest of the cube very smooth after EOLine: F2L can be solved with R, U, L (sometimes D) and without cube rotations. It also benefits F2L lookahead and one-handed solving. Its biggest weakness is that EOLine is unintuitive and hard to plan. The solver must recognize bad edges, orient them, and track the DF/DB line during inspection. Additionally, a smooth ZZF2L solving requires plenty of extra skills that are hard for inexperienced ZZ solvers.

A standard ZZ solve has three main stages: EOLine → ZZF2L → Last Layer.

The last layer can be solved with either ZBLL (493 algorithms), COLL + EPLL (VH Method), or OCLL + PLL (also good, especially for Sune cases).

EOLine

EOLine means solving edge orientation and placing the two line-edges, DF and DB, in one step. These two edges form a line on the D face. In ZZ terminology, a bad edge is a misoriented edge, and an oriented edge is one that can later be solved without needing F or B quarter turns. In conventional ZZ solving, we use white/yellow top and green/blue front.

Edge Orientation Inspection

In the first cube, all visible edges are correctly oriented; in the second cube, all visible edges are misoriented (bad edges). There must be even number of unoriented (bad) edges.

Here are some rules to identify a bad edge (assume the cube is held with white/yellow on top and green/blue in front):

First look at the U and D face (8 edges).

(1) Orange/red → The edge misoriented (bad);
(2) White/yellow → The edge is oriented;
(3) Green/blue → Inspect the color on the edge. If it's white/yellow, it's bad; otherwise, it is oriented.

Then look at the front and back faces of the E-slice (4 edges). The exact same rules apply.

Edge Orientation Strategies

The basic EO idea is that an F or B quarter turn changes the orientation of the four edges on that face, while F2, B2, and any other turns do not affect edge orientation. Most EO solutions work by gathering bad edges onto the F or B face using R, L, U, D moves, then turning that face to fix them.

# of bad edges	Probability	Strategy
0	0.05%	Done
2	3.22%	Move one to F/B and make a turn, swap it with the other misoriented one without disturbing any other F/B edges, and make a final F/B turn
4	24.17%	Gather all edges onto F/B and make a turn.
6	45.12%	Move 3 misoriented edges to F/B and make a turn. Then gather the new misoriented edges and 3 old misoriented edge
8	24.17%	Treat as 4 + 4
10	3.22%	Treat as 4 + 4 + 2. Keep track of the 2 good edges rather than the 10 bad edges initially
12	0.05%	Treat as 4 + 4 + 4

Other Tips:

1. Start with the F/B face that already has the most bad edges.
2. Avoid moving one bad edge into F/B if it pushes another bad edge out.
3. For 6+ bad edges, usually use both F and B faces.
4. Try to place bad edges into F and B at the same time when possible.
5. F/B turns with only two opposite bad edges do not fix EO, but may help set up the line.

Combining EO and Line

Once EO becomes familiar, try to combine EO and Line placement into one step: EOLine. Start with easy cases, such as 2 or 4 bad edges, then move to harder cases as your recognition improves.

The key is not to plan every line move separately, but to predict where the DF and DB edges will end up after EO. During EO, mentally track the two line-edges. If possible, choose EO moves that do not disturb one of them, especially when a line edge is already in the D-layer.

For an efficient EOLine, try to make the two line-edges end in positions that are easy to solve:

- Best: DF and DB are already solved.
- Also good: the line is in DL/DR and can be finished with one D or D’ move.
- Avoid: one line edge is easy to solve, but the other is far away or awkward.

In practice, try to move both line edges into easy positions during EO. Sometimes it is worth using one extra move if it gives better fingertricks or a smoother transition into F2L.

Solving Examples

Scramble 1: F U2 D2 B2 U' F U' R' L' D R'

Analysis: There are 4 bad edges on UF, UL, DL and DB. First observe that D' would both solve the yellow-green edge and move the UL edge to the B face. Then we can perform L' to set the yellow-blue edge. After that, place the other two edges. The final B step would solve the yellow-blue edge.

Solution: **D' L' R' U2 B**

Scramble 2: F' B' U L2 D B R2 L2 U'

Analysis: There are 8 bad edges. Four bad edges are already on the F face and three of them are already on the B face. We can solve them first and then solve the yellow-green edge.

Solution: **F D B U2 F2**

Scramble 3: R F2 B' D' L U2 L2 D2 B

Analysis: There are 6 bad edges: UF, UR, DL, DB, BL, BR. Three of them are already on the B face. Aim at doing 3 + 3 both on B face. Use B' U2 to bring the yellow-green edge out as well as moving UF to B face. Use D' to set the DL edge. Finally, use L' to set the yellow-blue edge and solve it. A 1×2×2 block can be solved simultaneously.

Solution: **B' U2 D' L' B' L B2 F2**

ZZF2L

This stage completes the first two layers by building two 1×2×3 blocks on either side of the Line made in the previous stage. Because all edges are now oriented, it is possible to complete this stage using just R, U and L moves.

In ZZF2L, we solve left and right side separately, but the process can be overlapped. For one side, we first solve a 1×2×2 block, and then solve a 1×1×2 block like the Fridrich last slot. In most cases, the solving order is: Left 1×2×2 → Right 1×1×2 → Left 1×2×2 → Right 1×1×2.

To build a 1×2×2 block, you need one D-layer corner, one D-layer edge, and one middle-layer edge. Usually, start by connecting the corner with one of the two edges to make a 1×1×2 block on the U layer. After that, place the remaining edge into the correct middle-layer position (FL, BL, FR, or BR) and connect it to the block with a U move.

A 1×1×2 block is usually built similar to CFOP F2L. ZZ mainly relies on L, R, and U moves but no cube rotations. If the pieces are not in the U layer, bring them up first. If they are already connected incorrectly, separate them, then reconnect them in the correct way.

ZZF2L lookahead tips:

Because ZZ uses no cube rotations, some F2L pieces may be partially hidden. Good lookahead depends on tracking pieces without constantly checking the back or bottom faces.

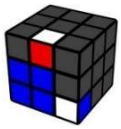
- Use elimination to identify hidden edges and corners.
- During EOLine, start looking for the first F2L pieces instead of checking the line.
- Be flexible: build blocks using either the D-layer edge or the E-slice edge first.
- When possible, place early blocks in the back to make it easier to see the front.
- Avoid scanning D/B faces; use R/L moves to bring hidden pieces into view.

ZZF2L Algorithms

Final Moves for intuitive solving:

R

U


R

U

U' R

U R2'

D-Layer 1×1×2

R2 U R2'

U2 R' U2'

U R U'

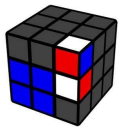
U' R

R2 U R' U

R U

R U2 R U'

U' R' U R' U'


U' R' U R U'

R U' R' U2

R U2 R2 U'

U' R U R2

R U R' U' R

U' R U R' U

Badly Connected Blocks

R’ U’ R U R2 U’ R’

R’ U2 R’ U2 R U R

L’ U R’ U’ L U2 R’ U R’ U2 R

R U2 R2 U’ R’

R2 D’ R U R’ D

R U2 R’ U’ R U R2

R U R’ U’ R2 U R2

R U’ R’ U2 R U2 R2

Multiblocking

R U R2 U2 R’

U’ R’ U2 R U R’ U’ R2 U R

U’ R’ U2 R U R’ U’ R2 U R

R U’ R’ U’ R’ U R

U’ R’ U’ R U2 R2 U2 R’

R’ U’ R’ U’ R’

U R’ U2 R2 U2 R U R’

U’ R2 U D R’ U2 R D’ R

U2 R’ U R’ D R’ U2 R D’

U L’ U R’ U’ R’ L U2 R’

R’ U2 L U’ R’ U L’ U R’

U R’ U2 R’

U’ R2 D R’ U2 R D’ R

U’ R2 U2 R2’ U R U’ R’

Petrus Method

The **Petrus Method** was created by Lars Petrus in 1981. This method uses very intuitive solving and cleverly integrates edge orientation into the F2L process. It requires far fewer algorithms than standard F2L and ZZF2L, and leads to the same last layer pattern as ZB. However, it can be sometimes hard to optimize block building, plus it is difficult to be consistent during speedsolving. As a result, Petrus has become less popular in speedcubing but remains its place in FMC solving.

A standard Petrus solve has the structure:
2×2×2 block → 2×2×3 block → Fix bad edges → Finish F2L → Last layer (ZBLL)

Build a 2×2×2 Block

The first step is to build a 2×2×2 block anywhere on the cube. This means solving one corner and the three edges around it. It is usually done intuitively by connecting small blocks: first a 1×1×2, then a 1×2×2, and finally the full 2×2×2 block. Lars Petrus recommends choosing the easiest starting corner during inspection rather than always forcing the same color.

1×1×2 | 1×2×2 | | 2×2×2 |

R2

R D' R'

R2' D' R2

R' F R

D' R' D

Expand to a 2×2×3 block

After the 2×2×2 block is solved, expand it into a 2×2×3 block by adding one more corner and two more edges. At this point, the solver still has several free sides, so block-building is flexible. The solver usually has three possible corners to choose from, so picking the easiest expansion matters.

An advanced trick is to think ahead into EO while finishing the 2×2×3 block. Sometimes the last move of Step 2 and the first move of EO cancel, or a bad-edge pair can be fixed during the transition.

Fix Bad Edges (EO)

Orient the 7 remaining unsolved edges correctly. The goal is to make the rest of the F2L solvable using only the two free sides (R and U moves).

In the picture on the right, a bad edge is an edge that cannot be solved using only F and U moves. For white and yellow edges, that means the white/yellow sticker is not facing up/down. We need to use some R and L moves to orient them.

In the following algorithms, light grey pieces are bad edges. Bad edges can only occur in pairs.

R U R’

R U’ R’

F R U R’

F’ R U R’

L R’ F L’ R2 U R’

R’ F R

R U2’ R’

M’ U M

F R U2 R2’ F R

(r U’ r) (R U’ R’)

Finish F2L

Finish F2L intuitively using only R, U moves (D is also available if necessary). After that, last-layer edges will be oriented automatically. You can apply either ZBLL algorithms (for advanced solvers), COLL + EPLL (VH Method), or OCLL + PLL (also a good choice, especially for Sune cases).

Solving Example

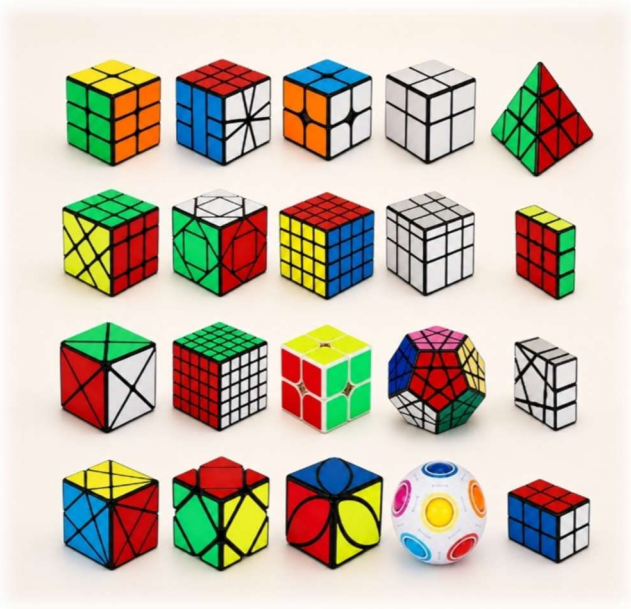
This is a real lucky solve using the Petrus Method.

29 HTM / 3.07s = 9.45 TPS

Scramble: F D B’ R2 D’ R D’ L’ F2 R’ F2 U2 L U2 D2 L B

y //Inspection
D’ L’ //2×2×2
U R’ F’ R2 F2 //2×2×3
R’ F’ U2 F //EO
U R’ U’ R D R’ U R D’ //ZZF2L
U2 F’ r U R’ U’ r’ F R //ZBLL-L

4 Other Events



Welcome to the wider world of cubing! This chapter introduces big cubes, twisty puzzles, and their solving methods. It covers 4×4×4 through 7×7×7, Pyraminx, Skewb, Square-1, FTO, and Megaminx. You will learn the basic techniques for solving larger cubes, handling parity, and understanding different puzzle structures.

Prerequisite: Sub 25s solving time in 3×3×3 cube
(except Pyraminx, Skewb, and FTO)

4.1 Big Cubes

Big cubes extend the challenge of the standard 3×3×3 cube by adding more pieces, larger centers, and longer edge groups. They are interesting because they require both familiar 3×3×3 solving ideas and new techniques such as center building and edge pairing. As the cube size increases, solving becomes more challenging and requires better organization, lookahead, and efficiency.

This tutorial introduces the basic techniques used in big cube speedcubing. It covers how to solve centers, pair edges, reduce the cube to a 3×3×3, handle common parity cases, and improve efficiency on 4×4×4 through 7×7×7 cubes.

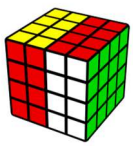
	2×2×2	**3×3×3**	**4×4×4**	**5×5×5**	**6×6×6**	**7×7×7**
Center Pieces	0	6	24	54	96	150
Edge Pieces	0	12	24	36	48	60
Corners	8	8	8	8	8	8
Total Pieces	8	26	56	98	152	218
Parity Cases	None	None	OP, PP	OP	OP, PP	OP
Average WR (July 2026)	0.86 s (ao5)	3.51 s (ao5)	18.56 s (ao5)	33.73 (ao5)	64.94 (mo3)	96.86 (mo3)

Notation

Larger cubes don't have a universal notation, but we generally use **nR(w)** for rotating n layers of the right side (R). For example, you can have R, 2R (or Rw), 3R (or 3Rw), etc. The lowercase r often refers to Rw as in this tutorial. An algorithm involving both Rw and r in the 3×3×3 cube would usually distinguish them. Additionally, in this tutorial, m, e, and s move refer to rotating the very middle slices, while M, E, and S represent 3×3×3 moves.

R
Outer layer

Rw or **r**
Wide turn – 2 layers

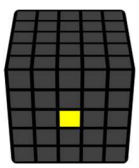


MR
Inner Slice

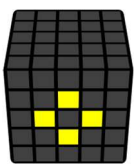
3Rw or **3r**
Wide turn – 3 layers

We usually call the pieces in a big cube like this:

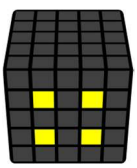
Middle Center



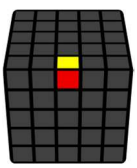
Edge Center



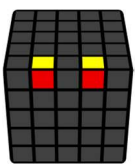
Corner Center



Midge



Wing

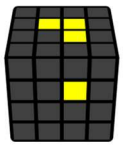


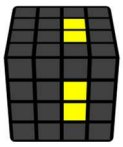
4×4×4 Cube

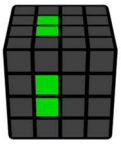
The **Yau Method** is the most popular method for 4×4×4 and 5×5×5 cubes because of its high efficiency. Unlike the 3×3×3 CFOP Method, solving a higher-order cube by a better method doesn't mean more algorithms, but it pays attention to the efficiency of the process. Considering this, I recommend big cube beginners (with a sub 25 level in 3×3×3 cube) to start learning the Yau Method directly for 4×4×4 and 5×5×5 cubes.

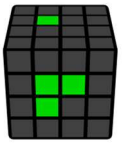
F2C (Two Opposite Centers)

The 4×4×4 doesn't have fixed centers, but their locations must be correct when the cube is solved. This means you need to place the centers in the correct order. Finish F2C as fast as you can.


r U' r'


r U2' r'


r U2' r'


r U r'

Correct

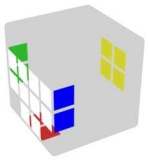
W3E (Three White Edges)

Hold the white centers on the left (L). Place a white edge pair on the 2L and 2R slices using outer layer moves, as shown in the first two pictures. Pair them with r moves and place them on L. Repeat this process for two more white edge pairs, ensuring the cross edges are solved in the correct relative positions.

r [U]

r2 [U']

1 solved


3 solved

Examples of pairing edges:

r U2' r2'

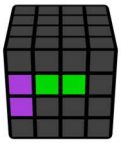
l U l

r U' r2'

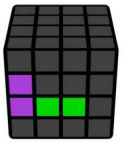
R' F r2'

L4C (Last Four Centers)

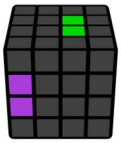
Solve L4C using r, 3r, U, and F moves, keeping the white centers on L and the unsolved cross slot in UL. Ensure the centers are in their correct relative positions. We usually place the center we are solving in F. Switch to the center above it next. The general idea is shown below, with the unbreakable edge highlighted in purple.



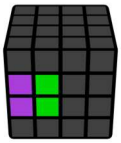
3r U' 3r'



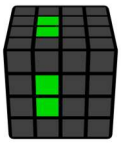
3r U 3r'



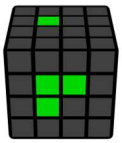
ML' U2 ML



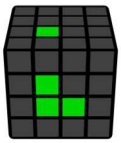
First Pair Solved



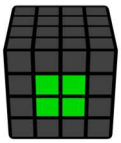
r U2' r'



r U r'



r U' r'



Second Pair Solved

LCE (Last Cross Edge)

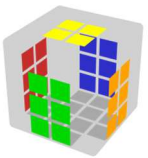
Place the last cross edges on the E slice, pair them using the **Flip Algorithm,** and then position them correctly on the D layer.

Flip Algorithm: R U R' F R' F' R or **R' F R F' R U' R'** (recommended)

u' (R U R') u

u' (Flip) u

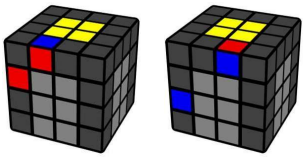
u2 (R U R') u2



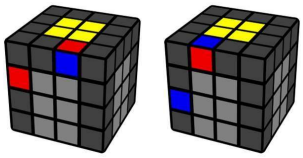
Solved

L8E (Last Eight Edges)

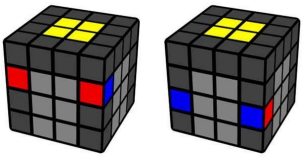
L8E offers some flexibility. The 3-2-3 method (referring to the edges) is an efficient method for solving it. There is a real example below. Start by focusing on efficient edge pairing below. Place the edge pairs interlaced, then use U moves to solve them.



R U' R' / U2 R U R'



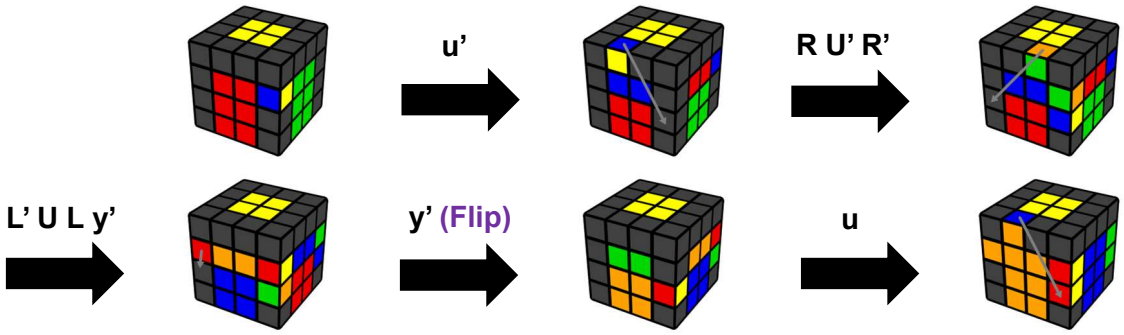
F R' F' R / U F' U' F / U' F' U F



(Flip)

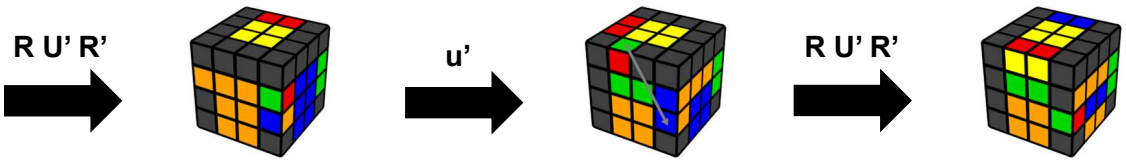
First Three Edges

Perform u', then place 3 edge pairs on the E slice so that when you do u, 3 edge pairs are formed.



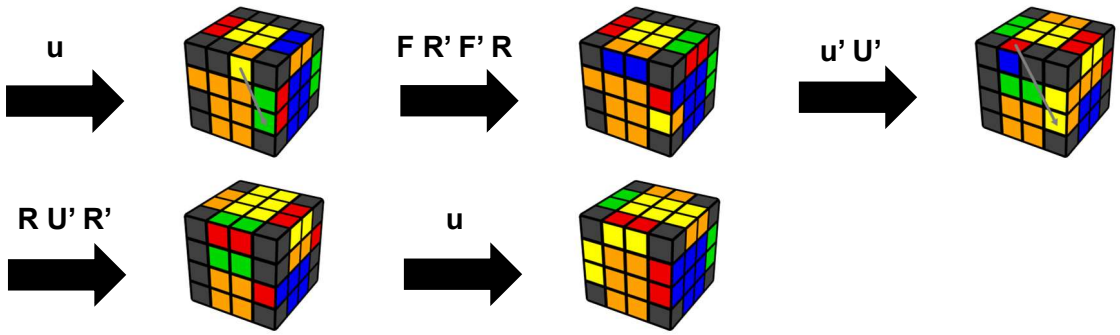
Next two Edges

After this, use 2 of the F2L slots to pair up the next 2 edges. You can adjust your method for solving the last 5 edges based on the situation. You can also follow the simplest way, just using the algorithms above.



Last Three Edges

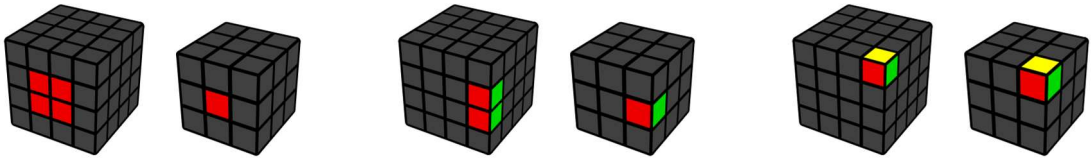
When solving the 10th edge, pay attention to the position of the last two edges.



3×3×3 Stage

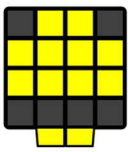
Solve the cube as if it is a 3×3×3 cube. Be aware of two possible parity cases that don't occur on a 3×3×3 cube. Parity cases occur on a 4×4×4 cube because it is actually an even permutation, meaning that 2 pairs of edges are swapped, or 2 edges are flipped.

Reduction



EO Parity

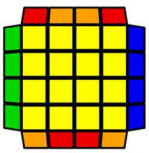
odd number of flipped edges



r U2 x r U2 (r U2' r' U2) l U2 (r' U2' r U2) r' U2' r'

EP Parity

odd number of edge pairs swap



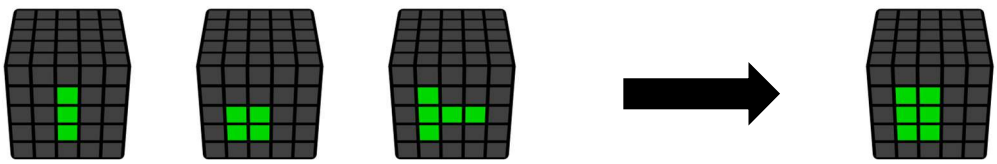
MR2 U2 MR2 u2 MR2 u2 [U2]

5×5×5 Cube

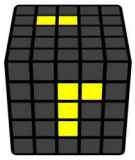
The 5×5×5 cube is not significantly harder than the 4×4×4 cube. In fact, it has fixed centers and doesn't have EP parity cases. The Yau method can also be applied to solve a 5×5×5 cube, but there is a major difference: L8E. The efficient 3-2-3 method in the 4×4×4 cube doesn't apply to a 5×5×5 cube, so we can use part of the 8-4 method.

F2C (Two Opposite Centers)

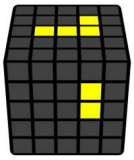
Solving the centers of a 5×5×5 cube is more complex. To solve a center, you can start by forming a centerline, a 2×2 block, or a T-shape. Once you have one of these, expand it into a 2×3 block. Finally, complete the center by solving the last line.



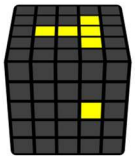
Examples of solving the first two centers:



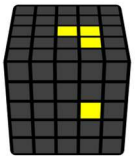
r U r' *(2x3)*



(r' U r U') (r U' r') *(2x3)*



r U(') r' *(2x3)*



r U r' *(2x2)*

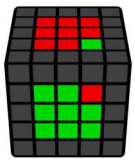
W3E (Three White Edges)

This step is similar to a 4×4×4 cube, but be careful with the orientation of the midges. It might take some time and practice to find the midge and the wings of one pair quickly.

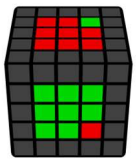
L4C (Last Four Centers)

Solve similarly to what F2C introduced. Follow the same sequence as the 4×4×4 cube. Be proficient in turning the F layer. For instance, we can also apply **x' z2 l' F(')** for the third example of solving the first two centers above.

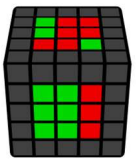
Examples of solving the last two centers:



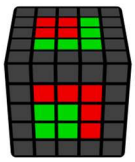
(r U r' U) (r U2' r')



(r U' r' U') (r U2' r')



(r U' r' U) (r U2' r')



(r' F r U) (r U2' r')

LCE (Last Cross Edge)

Pair up the last white edge and put it into the right spot.

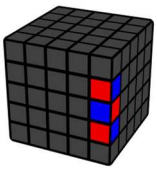
F4E (First 4 Edges)

On a 5×5×5 cube, each edge is made of three pieces. Use the three algorithms to place the edge pieces into the E-slice slots, then use Uw or 3Uw moves to pair them. Once an edge is complete, move it to the U layer.

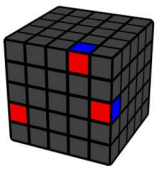
Do not worry about the centers yet; the algorithms use inverse outer moves, so the centers can be fixed later. After pairing 4 edges, fix the centers. Before the next step, make sure at least 2 connected F2L edges are solved. Ignore the white and yellow colors in the picture below.

- **R U' R'** (Preserves edge orientation)
- **F R' F' R** (Changes edge orientation)
- **R' F R F' R U' R'** (Flip the edge)

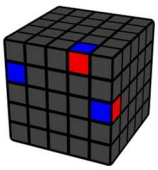
Examples of solutions during F4E:



e (R U R' F R' F' R) e'
(This also applies to L4E)



(R U' R' d) (R' U R e')



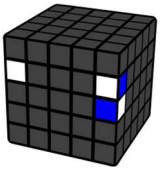
(F R' F' R) U' R' U R e'

2PF (First Two Pairs of F2L)

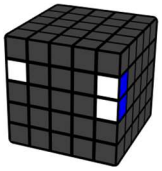
This step is not mandatory, but doing it would make your L4E solving more organized.

L4E (Last 4 Edges)

Use the last 2 slots in the e-slice to solve L4E. Insert the two parts of the edge into these two slots, then combine them correctly. If the 3 pieces are all separated, combine 2 of them first, as you did in the LCE step.

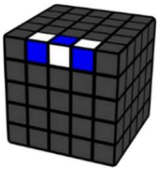


Uw' (R U R' F R' F' R) Uw



Dw' y' (R U R' F R' F' R) Dw

If you can't solve the last edge, you're facing a parity case. Ensure that the other 11 edges are solved before dealing with the parity case.



r U2 x r U2 (r U2' r' U2) Lw U2 (3Rw' U2' r U2) r' U2' r'

After that, it's the 3×3×3 stage without any parity cases.

6×6×6 and 7×7×7 Cube

If you have already mastered the 5×5×5 method, learning the 6×6×6 and 7×7×7 is not much harder, but there are a few new ideas to know.

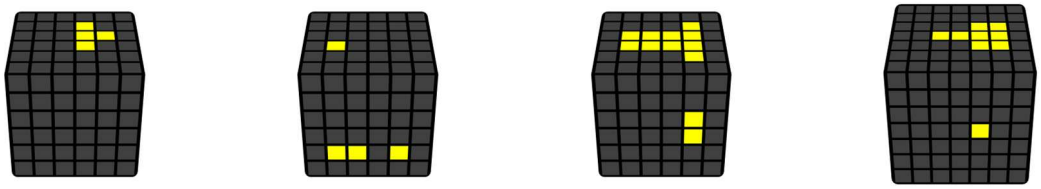
Most solvers use the **8-4 Method** for 6×6×6 and 7×7×7, which includes F2C, L4C, F8E, L4E, EP, and the 3×3×3 stage. Compared with the Yau Method, this avoids the problem of W3E interfering too much while solving the first four centers.

The 6×6×6 and 7×7×7 are the largest official WCA cubes, and they include special cases not seen on 5×5×5, such as some commutator-based center cases. For cubes larger than 7×7×7, centers can usually be solved in a similar way to the 7×7×7. The same notation will be used here.

Centers

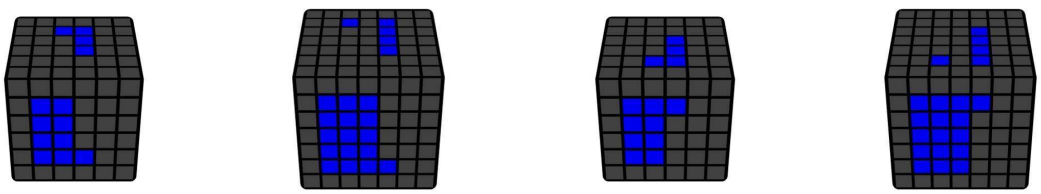
The typical approach to solving the first four centers of a large cube is simply solving line by line (make up a line in another face, then insert). Making grids or "T" shapes is also helpful. You can solve the first four centers intuitively if you have a 5×5×5 cube basis, but the last two centers are relatively harder, especially the commutator algorithm. Practicing helps a lot.

F4C



Rw U' Rw' (Line) | **3Rw 2Rw' U' 3mr'** (Line) | **Rw U Rw'** (4×3 block) | **3Rw U 3Rw'** (3×3 block)

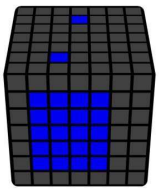
L2C Bad Cases for Inner Bar



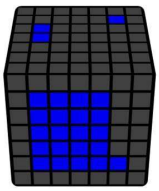
(3Rw U' 3Rw' U') (Rw U' Rw') (3Rw U2' 3Rw')

(3Rw U' 3Rw' U) (Rw U Rw') (3Rw U2 3Rw')

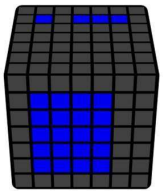
L2C Last Bar



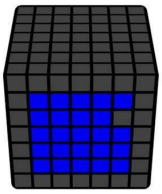
Attach one center wing using **U2 m' U2 m** (7×7×7 only)



Adjust the outer corner centers

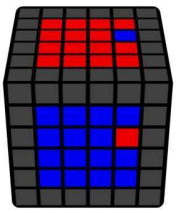


Make a partial bar and insert

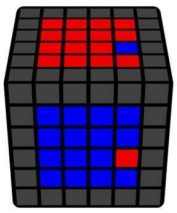


Apply the **Commutator Algorithm**

L2C Commutator Algorithms



Rw U 3rw U' Rw' U 3rw'



Rw U' 3rw U Rw' U' 3rw'

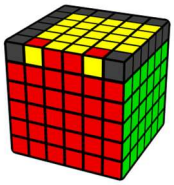
Edges

Mostly similar to F4E in the 5×5×5 cube, but F8E involves the D layer, which is a bit more complicated. Therefore, be proficient using R' D(') R moves to insert/take out edges.

A good solving order would be: Solve the 4 white edges, complete the cross, solve 4 middle-layer edges, complete two adjacent F2L slots, solve the rest 4 edges.

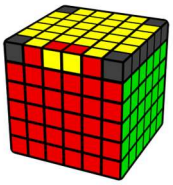
You can solve EO parity for larger cubes similarly to 4×4×4 cube or 5×5×5 cube. Sometimes you need to apply an inner parity and an outer parity.

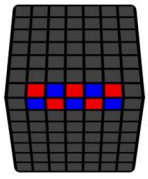
A 6×6×6 cube can have EP parities. You can solve it analogously to the 4×4×4 algorithm.



Outer Parity

r U2 x (r U2 r U2') (r' U2 l U2) (4r' U2' r U2) (r' U2' r')





Inner Parity / Big Parity

Outer Parity + 3R U2 x (3R U2 3R U2) (3R' U2 3L U2) (3/4R' U2 3R U2) (3R' U2 3R')

4.2 Twisty Puzzles

Twisty puzzles are puzzles that use different shapes, turning mechanisms, and piece structures to create new solving challenges. Although many of them rely on similar cubing ideas such as layers, permutation, orientation, each puzzle has its own unique logic and solving style.

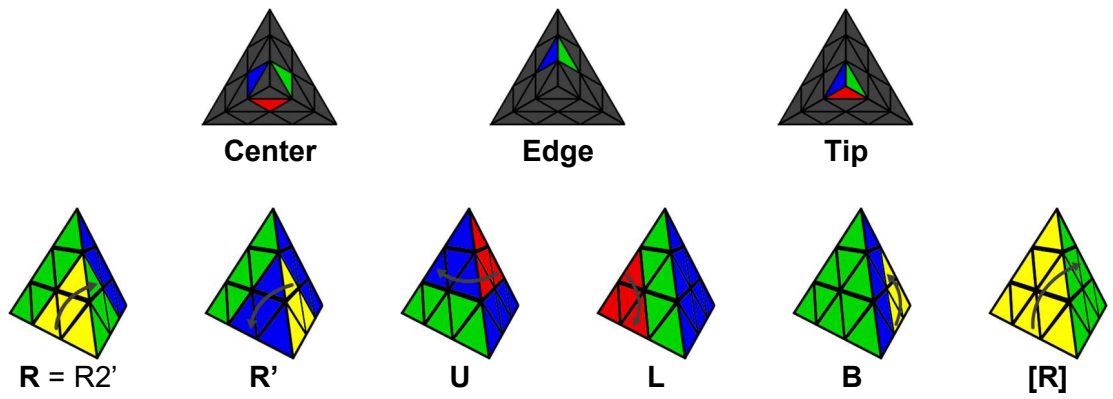
This tutorial focuses on the basic solving structure for five twisty puzzles: **Pyraminx, Skewb, Square-1, FTO, and Megaminx**. Four of which are current WCA events; FTO will become an official event in 2027. Pyraminx is fairly easy to pick up, Skewb has a unique but manageable structure, and Square-1 is one of the most challenging puzzles. FTO has a novel and special design, while Megaminx is closely related to the 3×3×3 in structure.

The following table shows the basic elements of each cube. The number of axles is equivalent to the number of faces for Pyraminx, FTO, and Megaminx.

	Pyraminx	**Skewb**	**SQ1**	**FTO**	**Megaminx**
Shape	Tetrahedron	Cube	Cube	Octahedron	Dodecahedron
Number of Axles*	4	4	3	8	12
Rotations per Axle	3	3	6/7/8/9/10; 2	3	5
Centers	12	6	0	24	12
Edges	6	0	8	12	30
Corners	4	8	8	6	20
Odd Parity	Yes	No	Yes	No	No
Difficulty	1	1.5	5	4	4.5
WR (ao5, by 2026)	1.14s	1.37 s	4.63 s	/	24.38 s

Pyraminx

Pyraminx is one of the easiest twisty puzzles to solve. Since the corners (often called tips) are not significant and centers are simple to solve, the puzzle is mainly focused on solving the edges.



Centers

Place the non-yellow tip on the top layer, then rotate the three yellow centers to the bottom face. The remaining centers will automatically align themselves.

R

R'

L

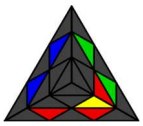
L'

B

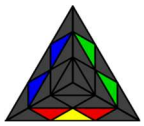
B'

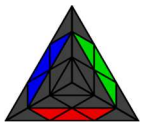
First Layer

Insert 3 yellow edges into the first layer.


U R U' R'

U' L' U L

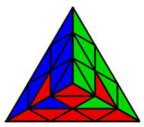

(R' L R L') (R U' R')


Solved

Last Layer

First, match the centers, apply **R U' R' U' R U' R'** 1~2 times until one edge is permuted.

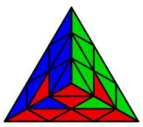
Place the edge oriented in B as shown, then apply **U R U' R' U L' U' L** to solve the whole cube.

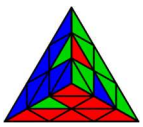


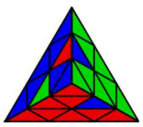
Last Layer Algorithms

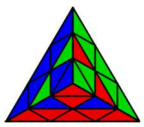
There are only 6 possible permutations of the last layer (including the solved state).

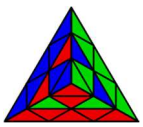
Two operations: **S = R' L R L'**, **H = L R' L' R**


H (U' R U R')


H (U R U' R')


S (U' L' U L)


R U' R' U'R U' R'


L' U L U L' U L

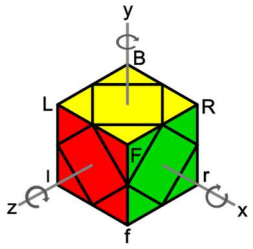
Skewb

A Skewb turns quite differently from a 3×3×3 cube, so the same logic doesn't apply. A great way to get familiar with it is by attempting to solve the first layer on your own. While doing this, pay attention to how the pieces move during each algorithm to better understand its mechanics.

Notation

In the picture on the right,

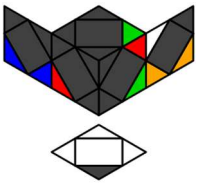
- The DBL corner is represented by the letter "b". Each letter on the corners indicates a 120° clockwise rotation with that specific corner as the center of rotation.
- The notations x, y, and z refer to rotating the entire cube 90° clockwise around the respective axes.
- Unlike the 3×3 cube, the Skewb has a unique way of holding the cube, as you're facing a corner.



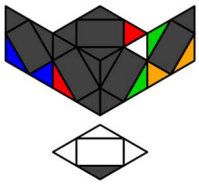
An important algorithm to know: **S = r' R r R'**

First Layer

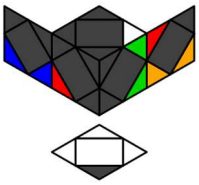
Each corner has a specific location, and when solving the white layer, ensure that the side colors are correctly paired as you rotate the corners into place. It's easy to position one corner at first, but then you must solve the corresponding corner next to it. To do this, move the unsolved corner to either UBR or UFL when its correct spot is in DFR, and then follow the algorithms provided below to complete the step.



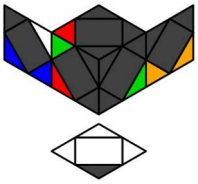
x R



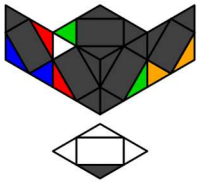
x r' R r



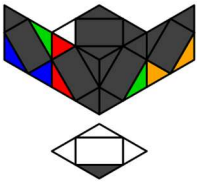
x z r' R r' R'



x R'



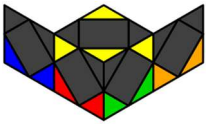
x z R r' R'



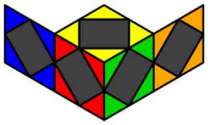
x z R' r' R

Top Layer

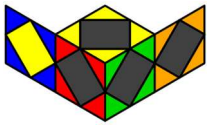
x z2 S



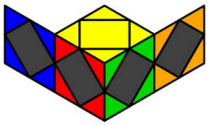
x S z' S



Solved

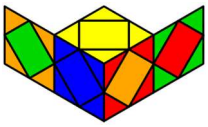


x S z2' S

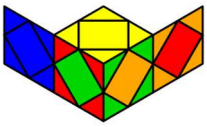


Solved

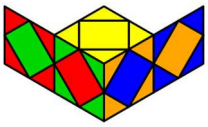
Side Centers

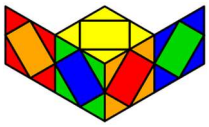


S z2' S



y' z2' S z2' S

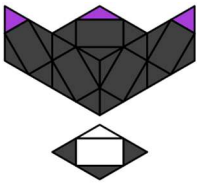




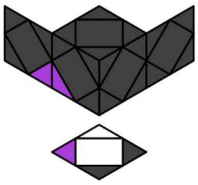
Apply S and transform to 1 or 2

Improving First Layer

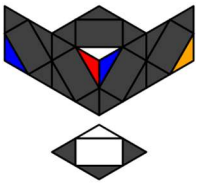
Try to solve the first two corners more efficiently. When solving the second corner, if the first corner is in BDL, use 1 move to bring it to UFR. Then, use 1-2 additional moves to solve it.



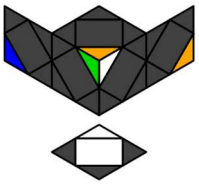
R' / L



L

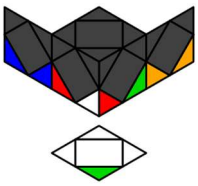


x z r R

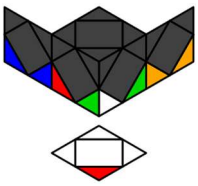


x R r'

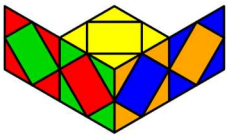
Algorithms



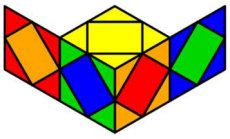
x z R r R' r'



x r' R' r R



(r l' B' l) (B l r l' r)



r R' r R' F' y l L' l

SQ1

Square-1 (SQ1) is one of the most challenging official twisty puzzles because its shape can change, making even the first step difficult for beginners. Its unusual structure also makes it especially interesting to explore.

After learning the basic turning mechanics, try creating shapes such as a **star** or **scallop**, challenge yourself to make the theoretical **M2** shape, and experiment with repeated move sequences to flip the equator or discover other transformations.

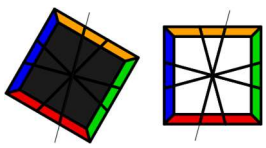
Notation

The SQ1 cube uses number notations to describe its movements. To properly hold the SQ1, ensure that the front-left (FL) side of the equator (middle layer) is the short edge. You should never rotate the entire cube; your left hand should always grip the short side of the equator. In the pictures, both the U (upper) and D (down) layers are shown in a top-down view.

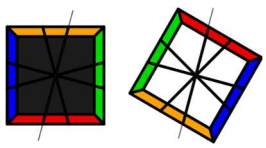
- The notation **±x ±y** refers to turning the U layer by ±30x° and the D layer by ±30y°.
- The symbol **/** represents a 180° rotation of the right side of the cube.

The bolded notations are official, while the non-bolded ones are simplified for easier memorization. I will provide 2 kinds of notation for most algorithms.

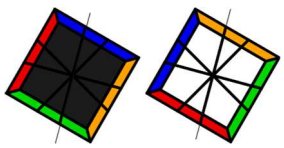
/ = R2; \ = R2'



(1, 0) = 1



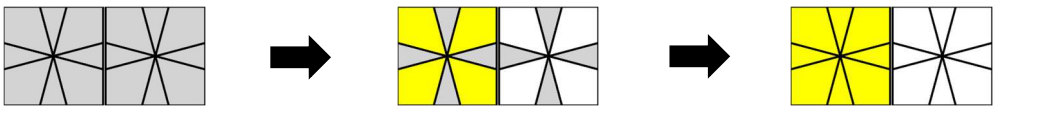
(0, 5) = 5



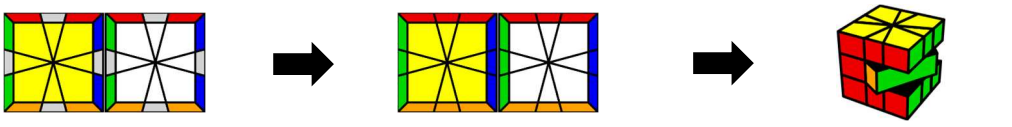
(4, -1) = 4-1

Process Overview

The **Lars Method** is widely used by both beginners and intermediate solvers. It involves 5+1 processes: (The +1 refers to Equator Flip)



Cube Shape (CS) → **Corner Orientation (CO)** → **Edge Orientation (EO)**

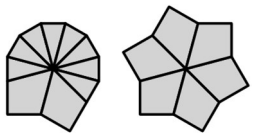


Corner Permutation (CP) → **Edge Permutation (EP)** → **Equator Flip (EF)**

CS

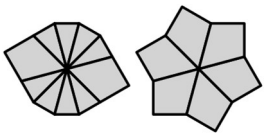
To create a 6-corner star on the D layer, follow these steps:

1. Pair three corners and move them to the left side of the D layer.
2. Pair the remaining three corners and move them to the right side of the D layer.
3. After these steps, you'll encounter one of five possible cases for arranging the corners.



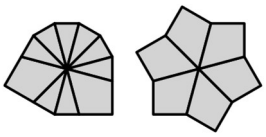
/ (2, 4) / (−2, −1) / (3, 3) /

/24/-2-1/UD/



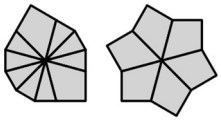
/ (2, 2) / (0, −1) / (3, 3) /

/22/-1/UD/



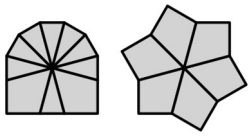
/ (−4, −2) / (−1, 4) / (−3, 0) /

/-4-2/-14/U'/



/ (4, 0) / (1, 2) / (3, −2) / (2, 1) / (0, 3) /

/4/12/3-2/21/D/

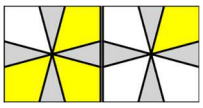


/ (2, 0) / (1, 2) / (3, −2) / (2, 1) / (0, 3) /

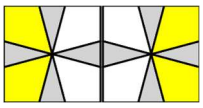
/2/12/3-2/21/D/

CO

The two algorithms on the right are designed to solve either 1 or 2 corners at a time. You can combine these algorithms to address the different cases you encounter while solving. Make sure the yellow face is oriented upward.



(1, 0) / (3, 0) / (−1, 0) **1/U/-1**

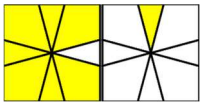


(1, 0) / (−1, 0) **1/-1**

EO

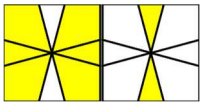
The same approach applies to CO. You can combine the basic algorithms to solve other cases.

M2 = (1, 0) / (−1, −1) / (0, 1)



(1, 0) / (3, 0) / (3, 0) / (−1, −1) / (−2, 1) / (−3, 0) / (−1, 0)

1/U/U/-1-1/-21/U'/-1

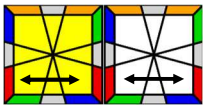


(1, 0) / (−1, −1) / (0, 1)

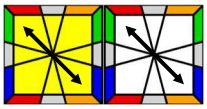
M2

CP

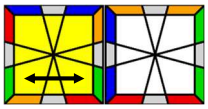
Solve the D layer first. If the U layer has the same case as D, then you can skip the next step. Otherwise, the position would be the adjacent corner swap except in one case: U layer opposite corner swap, apply the first algorithm twice to solve it.



/U'/UD/D'/



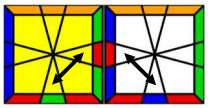
/UD'/U'D/



/UD'/U/U'/D/U'/

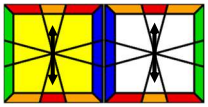
EP

There are over 100 EP positions. Combine the first two non-parity algorithms multiple times. If the cube remains unsolved, you're dealing with a parity case.



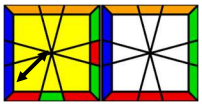
(1, 0) / (0, 3) / (−1, −1) / (1, −2) / (−1, 0)

1/D/-1-1/1-2/-1



(1, 0) / (−1, −1) / (−5, 1) / (−1, −1) / (0, 1)

M2 U2 M2



/ (−3, 0) / (0, 3) / (0, −3) / (0, 3) / (2, 0) / (0, 2) /
(−2, 0) / (4, 0) / (0, −2) / (0, 2) / (−1, 4) / (0, −3) / (0, 3)

/U'/D/D'/D/2/2/-2/4/-2/2/-14/D'/D

EF

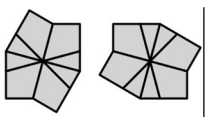
/U2'/U2'/U2'

SQ1 Intermediate

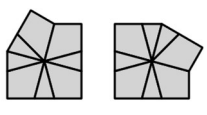
This part provides a set of useful formulas in the Intermediate Lars Method. It's meant for those who are willing to improve on it and achieve a 30s level.

CS

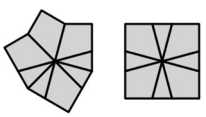
Here are some easy cases to solve. Try to be sensitive with these structures so you might get a chance to convert the case into the following shapes. Otherwise, focus on finishing the 6-corner-star quickly to save time.



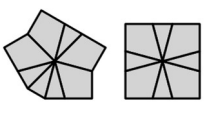
/33/



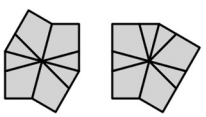
/3/



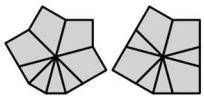
/-1/-3/



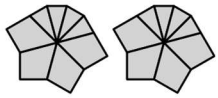
/2/3/



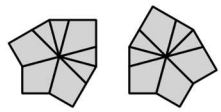
/-3/-3/



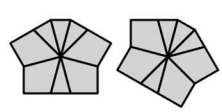
/-1-2/-3/



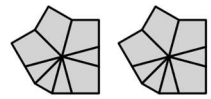
/12/-3-3/



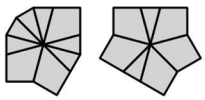
/1/-3-3/



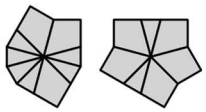
/-2/33/



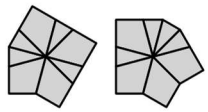
/-14/-3/



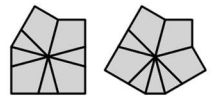
/23/12/3/-11



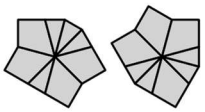
/-2-3/-1-2/-3/



/-2/-1/-3/

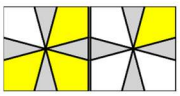


/01/0-1/-3/-11



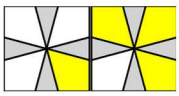
/3/-1-2/0-3/

CO (8)



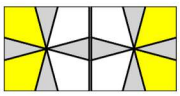
16/70

1/U/-1



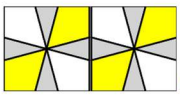
16/70

1/U2'D'/-1



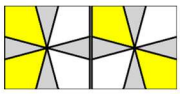
16/70

1/-1



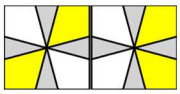
4/70

1/UD/-1



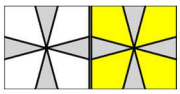
8/70

1/D/D/-1



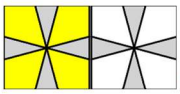
8/70

1/U'/U'/-1



1/70

1/U2'D2'/-1

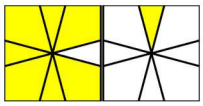


1/70

Solved

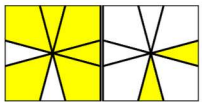
EO (8)

There are 8 cases in total. Learn 2 extra algorithms with a higher probability of occurring. Combine the first and the last algorithms to solve other cases.



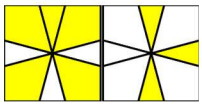
16/70

1/U/U/-1-1/-21/U'/-1



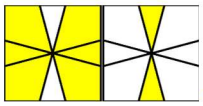
16/70

1/U/-1-1/U'/01



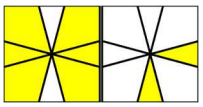
16/70

1/U/U/-1-1/U'/U'/01



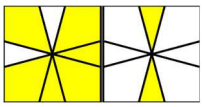
4/70

M2



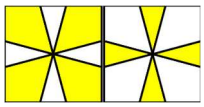
8/70

0-1/U/U'/11/U/U'/-1



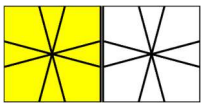
8/70

1/U/U/-1-1/-2
/-4-1/01



1/70

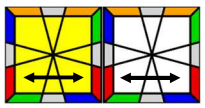
1/-1-1/-24/-1-1/01



1/70

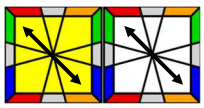
Solved

CP (8)



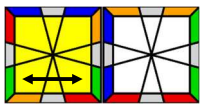
16/36

/U'/UD/D'/



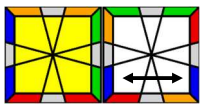
1/36

/UD'/U'D/



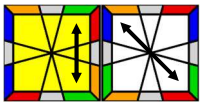
4/36

/UD'/U/U'/D/U'/

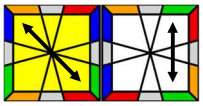


4/36

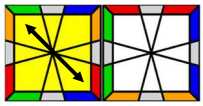
/UD'/D/U'/U/U'/


4/36

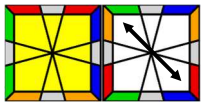
/D’/D/D’/D/


4/36

/U’/U/U’/U/


1/36

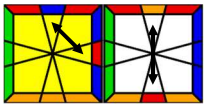
/U/U’/D/D/U’/U/


1/36

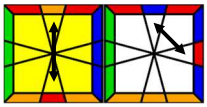
/D’/D/U’/U’/D/D’/

EP (6)

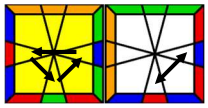
Here are some useful EP algorithms that can’t be solved by combining the basic algorithms. These specialized algorithms will help in more complex situations.



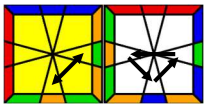
1/U/U’/2-1/11/U’/U/U’/-1



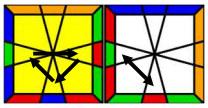
-1/D’/D/1-2/-1-1/D/D’/D/1



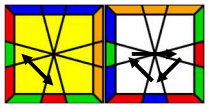
/UD/-5/2/4/4/4/2/-1/UD/



/UD/5/2/4/4/4/2/1/UD/



/U’D’/1/-2/-4/-4/-4/-2/5/U’D’/

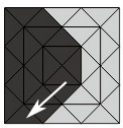


/U’D’/-1/-2/-4/-4/-4-2/
-5/U’D’/

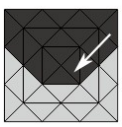
FTO

The FTO (Face-Turning Octahedron) was created later than other polyhedron cubes. It turns in a strange way: each turn impacts a large number of pieces. It has 4 pairs of opposite colors that never appear together in a single piece. Unlike other cubes, we can hold an FTO in two ways: with one face (F algorithms) or with one tip (T algorithms) facing ourselves.

R



L

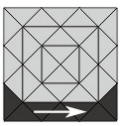


U

R

L

U



D

F

ML

MU

TU

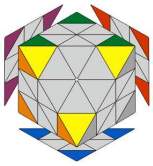
BR

BL

MR

ML

TR



Corners

Edges

Centers

BEN Method

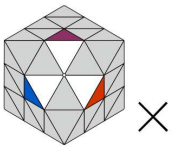
The layer-by layer solving process (called the **BEN Method**) of an FTO includes 3 stages:

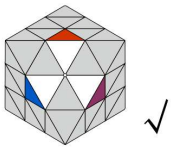
1. Bottom Layer: Edges → Centers → Corners
2. Middle Layer: Edges → Lower Centers → Upper Centers
3. Top Layer: Corner Orientation → Corner Permutation → Side Centers

Bottom Layer

Solving the first layer is very intuitive. It is a good practice to get familiar with how the cube works.

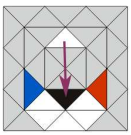
(1) Edges: The three edges need to be correctly ordered. There is no fixed center as reference.



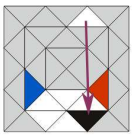


(2) Centers: Similar to solving the corner-centers in a 5×5×5 cube.

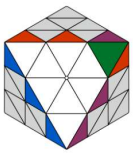
(3) Corners: Pay attention to the orientation when inserting them.



R U R' U R



R U' R' U R'



BR F' L

Middle Layer

Edges

Place the solved white face on the bottom. Look for the desired piece on the top layer and insert it. You can also use the mirrored left-hand algorithms. If there is no piece on the top, pull one out from the wrong slot.

R U R U' R'

R' U' R' U R

Lower Centers

Solve the 6 centers on the bottom. Then solve the 3 centers on the top. The first and the last algorithm are different because the first would scramble up the bottom centers.

R' L R L'

TU L D' L' MU'
L D L'

TU' R' D R MU
R' D' R

F R' L R ML'
R' L' R ML F'

Upper Centers

The yellow centers should be solved. We need to solve the 3 upper centers. If they are in the wrong slots, you can apply the algorithm to pull one out.

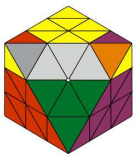
MR' U' R U MR U' R'

ML U L' U' ML' U L

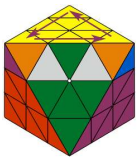
Top Layer

Corners

This step includes both corner orientation and permutation.



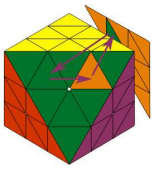
R U' R' U R' L R L'



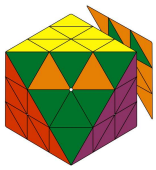
R U R' U R U R'

Side Centers

Only one 3-cycle algorithm is needed. You can utilize conjugate (i.e. setting moves) to solve more complicated cases or the mirror cases.



R U R' U' R' TR U R U' TR'



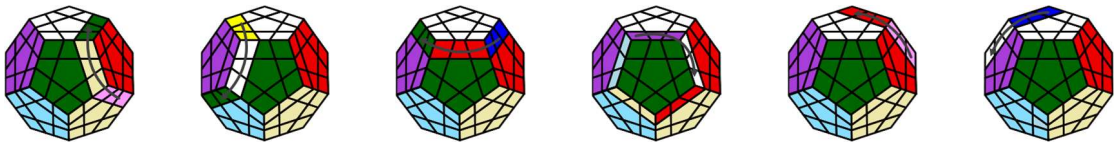
Alg BR Alg BR'

Megaminx

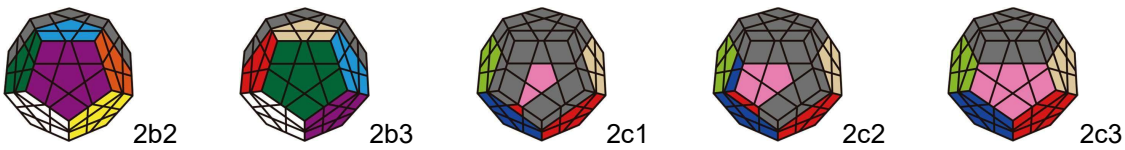
A Megaminx has a very similar mechanism to the 3×3×3 cube, but it has more faces and grater structural complexity. The mainstream method, called the **Westlund Method**, is also similar to the CFOP Method for the 3×3×3 cube.

The solving process can be classified as: F2L, S2L, and LL. The F2L and S2L process in Megaminx are very long, so it highly demands your 3×3×3 F2L solving skills.

Notation

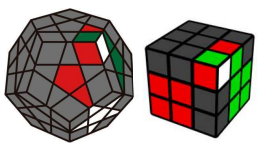


Steps for F2L and S2L

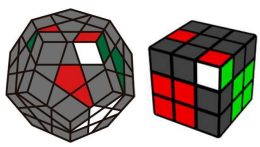


F2L (First Two Layers)

First, create a white star, similar to the cross on a 3×3×3 cube. Next, position the edge and corner pair above the corresponding slot. Then, complete five F2L pairs just like you would on a 3×3×3 cube. Here are some examples:



U' (R U' R' U) (R U R')



U' (R U2' R') U2 (R U' R')

S2L (Second Two Layers)

First 3 pairs

First, solve the last two yellow edges to complete the yellow star. Then, complete three pairs of S2L next to those edges. You can also choose a different color as your second "star."

Next 4 pairs

Extend the S2L by solving one edge and two corner-edge pairs. Repeat to complete one more block.

Last 3 pairs

First, solve the red-pink-blue block (shown above). Then, complete the last two pairs of S2L.

LL (Last Layer)

We have a 4-step process for solving the last layer, just like the one for the 3×3×3 cube. We only use algorithms for edges; we use a generic commutator-style method for solving the corners.

Edge Orientation

There are only three cases in edge orientation (EO). The third case is optional if you'd prefer to avoid learning too many algorithms. You can apply the first or second algorithm to simplify it into one of the easier cases.

F (R U R' U') F'

F (U R U' R') F'

*** F R U2 (R2' F R F') U2' F'**

Corner Orientation

There's an easy method to solve CO that we can learn.

First, position the top layer in **BL**.

Then, rotate the **BL** layer to move an **unoriented corner** into the upper back (**UB**) position and apply the algorithm. Afterward, rotate the BL again to bring the next unoriented corner into the UB position and repeat the algorithm. Continue this process (up to 5 times) until all corners are oriented. During this process, the S2L may become scrambled, but it will automatically be restored once all corners are correctly oriented.

Corner facing up

(U R U' R')2

Corner facing BR

(R U R' U')2

Example:

Scramble (Grey faces up): (R U2 R' U') (R U2 R' U2') (R U' R')

Solution: **z' y** */Rotation* **(U R U' R')2** */1st corner*

L' (U R U' R')2 */2nd corner* **L' (U R U' R')2** */3rd corner*

L' (U R U' R') */4th corner* **L' (R U R' U')2** */5th corner*

Edge Permutation

There are 5 cases in total. The two that are not shown are the inverse of the first two algorithms. You can solve any case by applying the first two algorithms one or two times. The third case is optional — if you prefer not to learn extra algorithms, you can solve it by combining the first two.

R2 U2' R2' U'
R2 U2' R2'

(R U R' F' R U R' U')
(R' F R2 U' R')

*** (L R U2) (L' U R')**
(L U' R U2) (L' U2 R')

Corner Permutation

A similar logic applies to CO. Repeat the process below. First, make grey faces FL.

Place an unpermuted corner in the UFL position and perform the sequence U' R' U to move the corner to DFR. Then, rotate the L face to position the correct slot for this corner in UFL. Apply U' R U to finish placing the first corner.

The second corner will now be in UFR. Align its correct location in UFL by rotating L, then perform U' R' U again. Rotate L to bring the third unoriented corner into UFL, and repeat U' R U to move it to UFR. Continue the process for the third corner, and the final corner will follow the same pattern as the first.

(U' R' U) L*m (U' R U)

(U' R' U) L*m (U' R U)

Example:

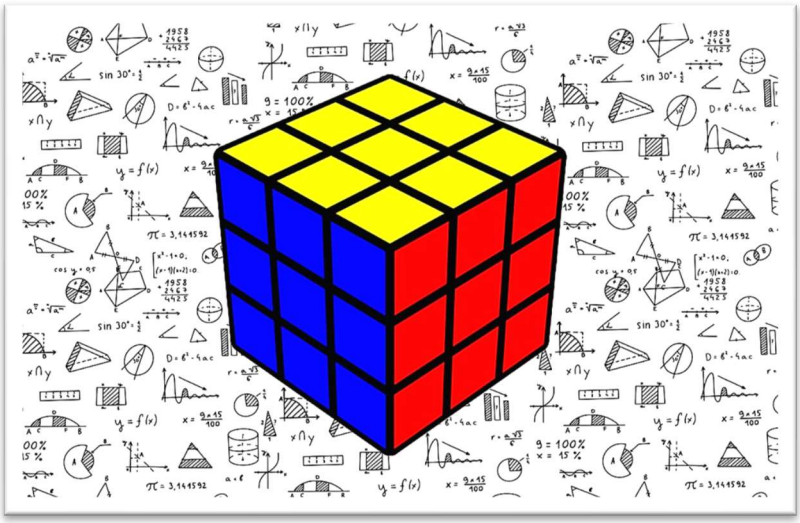
Scramble (Grey faces up): (R2 U2' R2' U') (R2 U R2' U')2 (R2 U2' R2')

Solution: **z'** */Rotation* **(U' R' U) L' (U' R U)** */1st corner*

L2' (U' R' U) */2nd corner* **L' (U' R U) L2 (U' R' U)** */3rd corner*

L2 (U' R U) */4th corner*

5 Cube Theory



In this chapter, we look deeply at the principles beyond the Rubik's Cube from the mathematical perspective. We will discuss topics such as group theory, legal states, invariants, Lagrange's Theorem, Domino Reduction, Cayley graph, God's Number, commutator, conjugation. In addition, this chapter covers the fundamental methods for 3BLD and 3FMC.

Prerequisite: Mastery of the CFOP Method
(Mastery of a high school algebra course would allow you to understand the content better)

Introduction: Why Cube Theory Matters

A Rubik's Cube is not only a toy; it is a mathematical system. Every move changes the cube in a controlled and reversible way. Because of this, the cube has hidden rules that cannot be broken by legal turns. For example, we have the three basic limitations on cube state:

1. One edge cannot be flipped by itself;
2. One corner cannot be twisted by itself;
3. Only one pair of pieces cannot be swapped by itself.

These facts may look mysterious at first, but they become natural once we describe the cube using groups, permutations, orientation, and parity.

Cube theory also helps explain why algorithms work. A PLL algorithm is not just a memorized sequence; it is a carefully designed permutation. A blindfold method is not just memorization; it is cycle decomposition. An FMC method is not only trial and error; it is a search inside smaller subgroups of the cube.

The purpose of this chapter is to build the mathematical language needed to understand these ideas. The goal is not to make the cube harder. Instead, the goal is to show that many "advanced" cube methods are based on a few simple mathematical principles.

5.1 Rubik's Cube as a Group

A ***group*** is a set G together with an operation $*$ satisfying:

i) Closure: $a, b \in G \Rightarrow a * b \in G$
ii) Associativity: $(a * b) * c = a * (b * c)$
iii) Existence of identity element: $\exists e \in G$ such that $e * a = a * e = a$
iv) Existence of inverse: $\forall a \in G,\ \exists a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$

A group has the following properties:

- The identity element, e (sometimes denoted as 1), is unique.
- If $ab = e$, then $a = b^{-1}$.
- If $ax = bx$, then $a = b$.
- $(ab)^{-1} = b^{-1}a^{-1}$
- $(a^{-1})^{-1} = a$

The following are some common examples of groups:

- The set of integers forms a group under addition. The identity element is 0, and the inverse of any integer a is its negative, $-a$.
- The set of $n \times n$ nonsingular matrices forms a group under multiplication. The identity element is I_n, and the inverse of any matrix A is its multiplicative inverse, A^{-1}.

For the Rubik's Cube, move sequences represent group elements. The operation is doing one sequence after another. The identity element e is the solved cube or the empty move sequence. The inverse of a move sequence is the sequence that undoes it.

We can write the Rubik's Cube group as $\mathcal{R} = \langle U, D, L, R, F, B \rangle$.

$\mathcal{R}$ is ***non-commutative***, as is the group of $n \times n$ nonsingular matrices, meaning that move order matters: $RU \neq UR$. If a sequence of moves is $A = a_1 \dots a_n$, then its inverse is $A^{-1} = {a_n}^{-1} \dots {a_1}^{-1}$. The order reverses because to undo several actions, we undo the last action first.

Permutation, Cycle, and Parity

A ***permutation*** is a rearrangement of a set. Formally, a permutation of a set X is a bijection

$$\sigma: X \to X$$

meaning every object goes to exactly one place, and every place receives exactly one object. On a Rubik's Cube, a move permutes pieces and stickers. If we track the moving stickers, the cube group can be viewed as a subgroup of a large symmetric group.

A ***cycle*** is a specific type of permutation where a subset of elements is shifted in a circular manner. For example, $(a\ b\ c)$ means $a \to b$, $b \to c$, $c \to a$. This is called a ***3-cycle***.

A swap of two elements, $(a\ b)$, is called a ***transposition***.

Every cycle can be written as a product of transpositions:

$$(a_1 \dots a_k) = (a_1 a_k)(a_1 a_{k-1}) \dots (a_1 a_2)$$

So, a k-cycle uses $k - 1$ transpositions.

The ***parity*** of a permutation tells whether it is even or odd:

$$\operatorname{sgn}(\sigma) = \begin{cases} +1, & \sigma \text{ is even} \\ -1, & \sigma \text{ is odd} \end{cases}$$

A permutation is considered ***even*** if it can be expressed as a product of an even number of transpositions; ***odd*** if it can be expressed as a product of an odd number of transpositions.

A 3-cycle is even. A 4-cycle is odd. The parity of a k-cycle is $\operatorname{sgn}(a_1 \dots a_k) = (-1)^{k-1}$.

On a cube, a face turn produces a 4-cycle of corners and a 4-cycle of edges. Each 4-cycle is odd, but together they make an even total permutation: $(-1)(-1) = +1$. This explains why an isolated swap (i.e. transposition) is impossible on a normal 3×3×3 cube.

Parity is like a balance rule. The cube allows swaps only when the imbalance is compensated somewhere else. To perform a transposition, another transposition must be done simultaneously.

5.2 Legal States and the Invariants

A cube state can be described by four pieces of data: $(\sigma_c, \omega_c, \sigma_e, \omega_e)$, where $\sigma_c =$ corner permutation, $\omega_c =$ corner orientation, $\sigma_e =$ edge permutation, $\omega_e =$ edge orientation.

There are 8 corners and 12 edges. At first, it seems that we may freely arrange and orient all of them. But the cube's turning mechanism imposes three restrictions.

Each edge k has 2 possible orientations: $e_k = 0, 1$, for $k = 1, \dots, 12$. Each corner l has 3 possible orientations: $c_l = 0, 1, 2$, for $l = 1, \dots, 8$.

Proposition (The 3 Invariants):

$$\text{(i)} \sum_{i=1}^{12} e_i \equiv 0 \pmod 2; \ \text{(ii)} \sum_{i=1}^{8} c_i \equiv 0 \pmod 3; \ \text{(iii)} \ \operatorname{sgn}(\sigma_e) = \operatorname{sgn}(\sigma_c).$$

These three propositions correspond to the three basic cube properties discussed in the introduction. Next, we will verify these propositions.

Proposition (i): An edge cannot flip by itself.

Proof: We encode $e_k = 0$ if edge k is oriented; $e_k = 1$ if edge k is flipped.

The total orientation is $e_1 + \cdots + e_{12} \pmod 2$.

In a solved cube, this sum is 0. Under common edge-orientation convention used in ZZ or DR theory, U, D, L, R moves do not change edge orientation, while F, B quarter turns change four edges. Changing four edges does not change the sum modulo 2.

Therefore, the number of flipped edges is always even: $\sum e_i \equiv 0 \pmod 2$.

A single flipped edge would give $\sum e_i \equiv 1 \pmod 2$. Contradiction. ■

Proposition (ii)**:** A corner cannot twist by itself.

Proof: Each corner has 3 possible orientations. Under ZZ or DR convention, we encode $c_l = 0$ if the white/yellow sticker of corner l faces up/down; $c_l = 1$ if the white/yellow sticker of corner l faces left/right; $c_l = 2$ if the white/yellow sticker of corner l faces front/back.

The total orientation is $c_1 + \cdots + c_8 \pmod 3$.

In a solved cube, this sum is 0. It is easy to verify that every legal operation U, D, L, R, F, B preserves this value, i.e. $\sum c_i \equiv 0 \pmod 3$.

A single twist of a corner would give $\sum c_i \equiv 1 \pmod 3$ or $\sum c_i \equiv 2 \pmod 3$. Contradiction. ■

Proposition (iii): One pair of pieces cannot swap alone.

Proof: In a solved cube, $\operatorname{sgn}(\sigma_e) = \operatorname{sgn}(\sigma_c) = 1$. Every legal operation U, D, L, R, F, B applies an odd permutation (precisely, a 4-cycle) to both σ_e and σ_c. Therefore, every legal operation multiplies both $\operatorname{sgn}(\sigma_e)$ and $\operatorname{sgn}(\sigma_c)$ by (-1). This implies that $\operatorname{sgn}(\sigma_e) = \operatorname{sgn}(\sigma_c)$ is always true.

If two pieces are swapped alone, a transposition, which is an odd permutation, is applied to exactly one of σ_e and σ_c. This operation multiplies exactly one of $\operatorname{sgn}(\sigma_e)$ and $\operatorname{sgn}(\sigma_c)$ by (-1), which violates the rule $\operatorname{sgn}(\sigma_e) = \operatorname{sgn}(\sigma_c)$. ■

This explains several common PLL facts: U Perm is a 3-cycle of edges; A Perm is a 3-cycle of corners, T Perm is one corner swap and one edge swap.

Counting legal cube states

The *size* of a group is the number of elements, or *cardinality*, in the underlying set of the group and is denoted $|G|$. For the Rubik's Cube, the size of $\mathcal{R}$ is the number of different legal states.

If the restrictions are ignored, we can count the number possibilities in each group $\sigma_c, \omega_c, \sigma_e, \omega_e$ and multiply them: $8! \times 3^8 \times 12! \times 2^{12}$.

The restriction says: One third of corner orientations are legal, one half of edge orientations are legal, and one half of piece permutations have matching parity. Therefore,

$$|\mathcal{R}| = \frac{8! \times 3^8 \times 12! \times 2^{12}}{3 \times 2 \times 2} = 43{,}252{,}003{,}274{,}489{,}856{,}000$$

This is approximately 4.3×10^{19}, or 43 quintillion legal states. If states that differ only by an overall cube rotation are identified as equivalent, divide by 24:

$$\frac{8! \times 3^8 \times 12! \times 2^{12}}{3 \times 2 \times 2 \times 24} = 1{,}802{,}166{,}803{,}103{,}744{,}000$$

If a cube is physically disassembled and randomly reassembled while keeping all pieces in cube, the probability that the cube is solvable under legal operations is

$$\frac{1}{12}$$

5.3 Group Concepts and Domino Reduction

Subgroup

A ***subgroup*** $H \subseteq G$ is a subset of G that forms a group under the same operation as G. It contains the identity and is closed under the group operation and inverses.

A ***generator set*** is a set of moves that produces a group. If S is a set of moves, then $\langle S \rangle$ means the group generated by all finite combinations of moves in S and their inverses.

If $S \subseteq \mathcal{R}$ is any subset of the group, then the subgroup H generated by S is the smallest subgroup of $\mathcal{R}$ that contains all the elements of S. For instance, $\langle U, D, L, R, F, B \rangle$ generates the full 3×3×3 cube group, while $\langle F \rangle$ generates a group that is a subgroup of $\mathcal{R}$ consisting of all possible cube permutations that can be obtained by rotating the front face.

Order

The ***order*** of an element g, denoted by $\mathrm{ord}(g)$, is the smallest positive integer n such that $g^n = e$.

For example, $\mathrm{ord}(R) = 4$ because $R^n = e$. $\mathrm{ord}(R\ U\ R'\ U') = 6$ because doing R U R' U' six times returns the cube to the solved state.

The order of other common sequences: $\mathrm{ord}(R\ U'\ R'\ U') = 5$, $\mathrm{ord}(R\ U) = 105$, $\mathrm{ord}(R\ U') = 63$.

Theorem: If the cube starts at the solved state, and one move sequence P is performed successively, then eventually the cube will return to its solved state.

Proof: The Rubik's Cube has only finitely many legal states. Therefore, in the sequence $e, P, P^2, ...$, some state must eventually repeat. Suppose $P^i = P^j$ for some $i < j$. Since every cube has an inverse, we can multiply both sides by P^{-i}: $P^{-i}P^i = P^{-i}P^j$, so $e = P^{j-i}$. Thus, after $j - i$ repetitions, the cube returns to the solved state. ■

Lagrange's Theorem

Now, we ask the question: if a set of moves generates a subgroup, how large is this subgroup? We first define cosets.

Let G be a group, and let $H \subseteq G$ be a subgroup. For any element $g \in G$, $gH = \{gh : h \in H\}$ is called a ***left coset*** of H, and $Hg = \{hg : h \in H\}$ is called a ***right coset*** of H.

Intuitively, a coset is a shifted copy of a subgroup. If H is a subgroup of G, then Hg is the same group moved to another location in the full group. The **Lagrange's Theorem** states that the size of any subgroup must divide the size of the whole group.

To prove this theorem, we first prove two lemmas.

Lemma 1: If $H \subseteq G$ is a finite subgroup that contains n elements, then any right coset of H contains n elements. In other words, every coset of H has the same number of elements as H.

Proof: For any element $g \in G$, $Hg = \{hg : h \in H\}$ defines the right coset. The map $h \mapsto hg$ from H to Hg is a bijection, so $|Hg| = |H|$. ■

Lemma 2: Two right cosets of a subgroup H are either identical or disjoint.

Proof: Suppose Hx and Hy have element in common, then for some h_1 and h_2, $h_1 x = h_2 y$. Then $x = {h_1}^{-1} h_2 y$, and some $h_3 = {h_1}^{-1} h_2$ gives $x = h_3 y$. So, every element of Hx can be written as an element of Hy: $hx = hh_3 y$ for every $h \in H$.

A similar argument shows the opposite: If one element in Hy, then every element of Hy can be written as an element of Hx. Therefore, if they have one element in common, they have every element in common and are identical. ■

We now can say that the right cosets of a subgroup ***partition*** the group into disjoint sets, and that each of these cosets contains the same number of elements.

Lagrange's Theorem: If G is finite and H is a subgroup of G, then $|H|$ divides $|G|$. Equivalently, $|G| = m|H|$ for some positive integer m.

Proof: The right cosets of $H \subseteq G$ partition G. Suppose there are m cosets of H in G. Each coset has the size of the number of elements in H, or $|H|$. Since $G = H_1 g \cup ... \cup H_n g$, its size is the sum of the sizes of all the cosets. Therefore, we can write $|G| = m|H|$. ■

Below is a list of some group generators and their sizes:

Generators	Size	Factorization / Ratio *m*
$\langle U \rangle$	4	2^2
$\langle U^2, R^2 \rangle$	12	$2^2 \cdot 3^1$
$\langle U^2, L^2, R^2 \rangle$	96	$2^5 \cdot 3^1$
$\langle U, R^2 \rangle$	14,400	$2^6 \cdot 3^2 \cdot 5^2$
$\langle U, R \rangle$	73,483,200	$2^6 \cdot 3^8 \cdot 5^2$
$\langle U, D, L^2, R^2, F^2, B^2 \rangle$ (Domino Reduction Group)	19,508,428,800	$m = 2{,}217{,}093{,}120$
$\langle U, D, L, R \rangle$	21,119,142,223,872,000	$m = 2048$
$\langle U, D, L, R, F \rangle$	43,252,003,274,489,856,000	$m = 1$
$\langle U, D, L, R, F, B \rangle$	43,252,003,274,489,856,000	$m = 1$

Domino Reduction

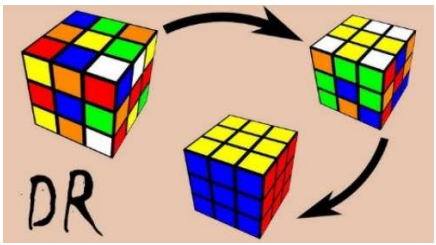
Domino Reduction, or DR, reduces the cube from the full move group $G_0 = \langle U, D, L, R, F, B \rangle$ to the subgroup $G_1 = \langle U, D, L^2, R^2, F^2, B^2 \rangle$. In this subgroup, U and D quarter turns are allowed, but the other four faces may only be turned by 180 degrees. The cube then behaves similarly to a 2×3×3 Domino Cube. DR is a common advanced method used in 3FMC (fewest moves) solving. It is considered the best human method to approach the God's Number.

A cube is in a DR state, relative to a chosen axis, when (1) all edges are oriented, (2) all corners are oriented, (3) the E-slice edges are in the E-slice.

Since $G_1 = \langle U, D, L^2, R^2, F^2, B^2 \rangle$ is a group, it satisfies closure. This means the three features mentioned above stay the same unless we use a forbidden turn.

Intuitively, DR shrinks the maze. The full cube has too many possible paths. By reducing to a subgroup, the remaining search becomes more structured. This is why DR is powerful in FMC.

A typical FMC path is: EO → DR (CO + Slice fix) → HTR or skeleton/insertions



DR also appears in computer-solving theory. Kociemba's Two-Phase Algorithm uses the subgroup $G_1 = \langle U, D, L^2, R^2, F^2, B^2 \rangle$ as its Phase 1 goal, and Phase 2 solves the cube from G_1.

For human FMC, the practical idea is:

1. Fix edge orientation (similar to the ZZ Method).
2. Set up a short DR trigger.
3. Enter the domino subgroup.
4. Finish using domino moves, HTR, block building, or insertions.

A **DR trigger** is a short sequence that completes the reduction after the bad edges and bad corners have been prepared. Common DR triggers can be R, R U R', R U' R', R U2 R', etc.

5.4 Cayley Graph and God's Number

Cayley Graph

A ***Cayley graph*** is a visual way to represent a group and its generators. Instead of only writing a group algebraically, we draw it as a network of states.

For a group G with generator set S, $G = \langle S \rangle$, the Cayley graph is built as follows:

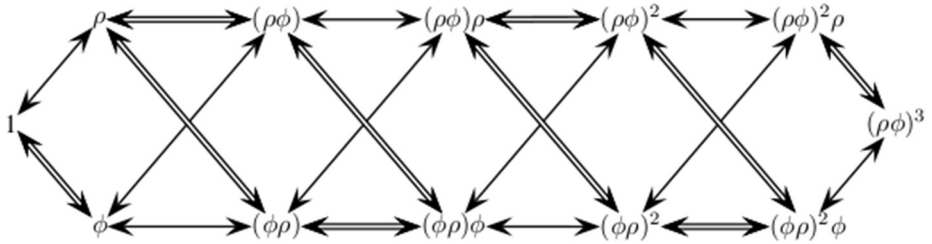
1. Each group element $g \in G$ is a vertex;
2. Each generator $s \in S$ gives a type of directed edge;
3. From each vertex g, draw an edge to gs because applying s changes state g into state gs.

In Rubik's Cube language, each vertex is a cube state, and each edge is one allowed move.

For example, the subgroup $\langle F \rangle$ has only four states: e, F, F^2, F^3. So, its Cayley graph is a 4-cycle (here 1 denotes the solved state):

$$1 \to F \to F^2 \to F^3 \to 1$$

The moves $\phi = F^2$ and $\rho = R^2$ generate the following graph (note that $\phi^2 = \rho^2 = 1$):



The thin arrows represent $\rho = R^2$, and the wide arrows represent $\phi = F^2$.

Obviously, we cannot draw the Cayley graph for the full Rubik's Cube group, but we can draw small graphs to help us see the structure of move sequences: which states are connected, how long cycles are, and how generators interact.

Cayley graphs also explain the idea of distance. The ***distance***, $d(g, h)$ is the minimum number of moves needed to get from state g to state h. The ***diameter*** of the graph is the maximum distance between the identity state and any state, $D = \max_{g \in \mathcal{R}} d(e, g)$.

In the Cayley graph above for $G = \langle \phi, \rho \rangle$, the distance between state 1 (solved state) and state $(\rho\phi)^3$ is 6, meaning that it takes at least 6 moves to get from state 1 to state $(\rho\phi)^3$. Since state $(\rho\phi)^3$ is the farthest from the identity state, the diameter of the graph is 6.

This is why Cayley graphs connect directly to God's Number. God's Number is the largest possible distance from the solved state to any other legal cube state in the Cayley graph of the Rubik's Cube group.

God's Number

As introduced in the last section, a cube state can be viewed as a vertex in a Cayley graph. Two states are connected if one move changes one state into the other. The distance $d = (g, h)$ is the minimum number of moves needed to get from state g to state h.

God's Algorithm is an ideal algorithm that always solves a cube in the fewest possible moves.

God's Number is the worst-case length of such an optimal solution:

$$\text{God's Number} = \max_{g \in \mathcal{R}} d(e, g)$$

where e is the solved state. Equivalently, it is the diameter of the Rubik's Cube graph.

The God's number depends on how we count moves:

Metric	Meaning	God's Number
HTM (Half-turn Metric)	Any outer face turn counts as 1 move	**20**
QTM (Quarter-turn Metric)	Only quarter turns count as 1 move	**26**
STM (Slice-turn Metric)	Any face or slice turn counts as 1 move	**18**

The God's Number for the Rubik's Cube is **20** under HTM, the most widely used metric. It was proved in July 2010 by a team of researchers: Tomas Rokicki, Morley Davidson, Herbert Kociemba, and John Dethridge.

The first known position proven to require 20 moves was the **superflip**, where all 12 edges are flipped. This provided a lower bound of the God's Number. However, proving that every single configuration requires *no more than* 20 moves is a monumental challenge. Researchers solved this through a "proof by exhaustion" relying on advanced computational mathematics.

This result is surprising because there are 4.3×10^{19} legal cube states, but none of them is more than 20 moves from solved. The cube is huge, but also very tightly connected.

Distance Distribution

God's Number describes the worst case, but most random cube states are not worst cases. The exact theoretical distribution of optimal HTM solution lengths is known only up to distance 15. For larger distances, simulations suggest that most random cube states are around 17 to 18 moves from solved. The simulated probabilities are:

$$P(16) \approx 0.0264,\ P(17) \approx 0.267,\ P(18) \approx 0.6704,\ P(19) \approx 0.03387$$

$P(20) > 0$, but probably below 10^{-11}. This is why "20" is the maximum, not the typical scramble difficulty. The 3FMC world record is 16 HTM single, 19.33 HTM mean.

Demigod's Number

God's Number 20 is exact, but proving it required a large computation. A **Demigod's Number** is a weaker bound that is easier to understand and reproduce.

A recent paper gives a high-confidence upper bound of 36 moves. This does not replace God's Number 20. Instead, it gives an easier probabilistic argument based on **sampling** and **graph theory**. The paper explains that the Rubik's Cube graph is *vertex-transitive*, and in vertex-transitive graphs the diameter can be bounded using the average distance between vertices.

Other Puzzles

The same idea applies to other twisty puzzles. Each puzzle has its own state graph, move set, metric, and God's Number. For small puzzles, we can sometimes compute the answer exactly. For large puzzles, even defining and bounding the answer becomes a serious mathematical challenge.

Some examples from a public table of known or bounded God's Numbers:

Puzzle	Metric	Known value or bound
2×2×2	HTM	11
2×2×2	QTM	14
Pyraminx	Without tips	11
Skewb	Standard metric	11
Square-1	Twist metric	13
Megaminx	FTM	Between 48 and 114

The important lesson is that God's Number is not just a Rubik's Cube fact. It is a graph-theory question: How far can any state be from solved?

5.5 Commutator and Conjugation

Definitions

An ***algorithm*** is a move sequence designed to achieve a specific effect. In cube theory, many algorithms are built from two structures: commutators and conjugates.

A ***commutator*** is $[A, B] = ABA^{-1}B^{-1}$. It measures how much A and B fail to commute. If $AB = BA$, then $[A, B] = 1$. This means a commutator does nothing when A and B commute.

On the cube, commutators are useful because they often affect only a few pieces. Most pieces moved by A are restored by A^{-1}, and most pieces moved by B are restored by B^{-1}. The remaining effect occurs near the overlap between A and B.

A key theorem in commutator is: If A and B affect localized sets of pieces and their supports overlap in exactly one piece, then $[A, B]$ cycles exactly three pieces and leaves the rest unchanged. This is called the **3-Cycle Commutator Theorem**, which will be discussed below.

A ***conjugate*** is $A : B = ABA^{-1}$. In cubing language, a conjugate is: setup → algorithms → undo setup. A is the setup, B is the main algorithm, and A^{-1} undoes the setup.

Conjugation lets us take an algorithm that works in one position and move its effect somewhere else. For example, if an algorithm B cycles three U-layer corners, but our target corners are elsewhere, we can use a setup A to bring the targets into those positions.

Many algorithms are combinations of both ideas: $A : [B, C] = ABCB^{-1}C^{-1}A^{-1}$. This is called a ***conjugated commutator***. It is especially important in blindfold solving and FMC insertions.

3-Cycle Commutator

We treat a cube move sequence as a function of stickers. If a sequence A moves sticker x to sticker y, we write $A\colon x \to y$ or equivalently, $Ax = y$.

Since cube moves are reversible, every move sequence is a bijection. This means no two different stickers can be sent to the same sticker.

We say that A ***affects*** element x if $Ax \neq x$. The ***support*** of A, written $\mathrm{supp}(A)$, is the set of all elements affected by A: $\mathrm{supp}(A) = \{x : Ax \neq x\}$. If $x \notin \mathrm{supp}(A)$, then $Ax = x$.

Theorem: Let $[A, B] = ABA^{-1}B^{-1}$ be a commutator. Suppose $\mathrm{supp}(A) \cap \mathrm{supp}(B) = \{x\}$, and $Ay = x$, $Bz = x$, where x, y, z are distinct elements. Then, $[A, B] = (y\ x\ z)$.

That is, if A and B affect exactly one common sticker x, A moves y to x, B moves z to x, then the commutator cycles exactly three stickers: $y \to x$, $x \to z$, $z \to y$.

Proof: Since $Ay = x$, $Bz = x$, we have $A^{-1}x = y$, $B^{-1}x = z$.

Track x: $x \xrightarrow{A} Ax \xrightarrow{B} Ax \xrightarrow{A^{-1}} x \xrightarrow{B^{-1}} z$. So $[A, B]x = z$.

Track y: $y \xrightarrow{A} x \xrightarrow{B} Bx \xrightarrow{A^{-1}} Bx \xrightarrow{B^{-1}} x$. So $[A, B]y = x$.

Track z: $z \xrightarrow{A} z \xrightarrow{B} x \xrightarrow{A^{-1}} y \xrightarrow{B^{-1}} y$. So $[A, B]z = y$.

Now consider any other element $u \notin \{x, y, z\}$. There are three cases:

(1) If u is affected only by A, then A moves it, B does nothing to it, A^{-1} moves it back, and B^{-1} does nothing to it. So u is fixed.
(2) If u is affected only by B, then A does nothing to it, B moves it, A^{-1} does nothing to it, and B^{-1} moves it back. So u is fixed.
(3) If u is affected by neither A nor B, then all four parts of the commutator leave it fixed.

Since $\text{supp}(A) \cap \text{supp}(B) = \{x\}$, there are no other possible cases. Therefore, the commutator affects only x, y, z, and it cycles them as $(y\ x\ z)$. Thus, $[A, B] = (y\ x\ z)$. ■

Based on the 3-cycle Commutator Theorem, a practical 3-cycle commutator often has two parts: interchange and insert. The ***interchange*** is usually one move that swaps or moves two target pieces in a simple way. The ***insert*** brings the third target piece, often called the ***odd piece***, into the correct interaction area. The insert is usually a conjugate.

Two common forms are [interchange, insert] and [interchange, insert']. Here, insert' means the inverse of insert, insert^{-1}. A basic procedure for deriving a 3-cycle is:

1. Find a simple interchange affecting two of them, and determine the odd piece.
2. Find the insert algorithm that inserts the third target into the overlap.
3. Write the commutator. Expand and cancel moves if necessary.

Here are some examples:

Example 1 (Corner): Scramble = R U' R' D' R U R' D

The interchange move is D, and the odd piece is ULF.
Method 1 = [interchange, insert] = [D', R U' R']
Method 2 = [insert, interchange'] = [B U2 B', D]

Example 2 (Edge): Scramble = F' U L' U2 F B' R2 F' B L U' F

The interchange move is M', and the odd piece is DR.
Method 1 = [interchange, insert] = [M', U R2 U']
Method 2 = [insert, interchange'] = [F R F', M]

Example 3 (Edge): Scramble = B2 R B R' L U' R' U R L' B

The interchange move is U, and the odd piece is BL.
Method 1 = [interchange, insert] = [U, F' E F]
Method 2 = [insert, interchange'] = [R' E2 R, U']

If there is no one-move interchange, a conjugated commutator is needed. We can perform a setup move to create an appropriate interchange move.

Example 4 (A Perm): Scramble: x' R' F R' B2 R F' R' B2 R2 x

Setup move = R2, then the interchange move is D2, the odd piece is UFR.
Method 1 = Setup: [interchange, insert] = R2: [D2, R U R'] = R2' D2 R U R' D2 R U' R
Method 2 does not work here because the interchange move is a double turn.

Example 5 (U Perm): Scramble: R' U R' U' R' U' R' U R U R2

Setup move = R', then the interchange move is U', the odd piece is FR.
Method 1 = Setup: [interchange, insert] = R': [U', L' E2 L]
Method 2 = Setup: [insert, interchange'] = R': [B' E B, U]

Here are some exercises:

Scramble	**Solution**
D' B' L' B' L F' L' B L F B D	[L, D R' D'] or [U' R' U, L']
U' B' U2 B' D' B U2 B' D B2 U	L2: [U', R D2 R'] or R2: [D', R U2 R']
L2 U' L2 U L2 R2 D' L2 D R2	[M2, F' U F] or [D' L2 D, M2]
R F' R' F' R F R F R F' R2	[L S2 L', F']
F' R' U2 F B' R2 B F' R F	R2: [F, D' S' D] or F: [E2, B L B']

Other Applications

Commutators and conjugations are very useful in Rubik's Cube solving: speedcubing, blindfolded, and fewest moves. You can memorize many OLL, PLL or Roux algorithms as commutators or conjugations. You can also use commutators to derive big cube edge algorithm.

OLL Algorithms

Scramble: F R' F' r U' R U' r', **Solution**: [L, F R' F'] = L F R' F' L' F R F' = **r U R' U' r' F R F'**

Scramble: R U R' U' M' U R U' R', **Solution**: [M', R U R' U'] = **r U R' U' M U R U' R'**

PLL Algorithms

If we know the algorithm (T) = (R U R' U') (R' F R2 U' R' U') (R U R' F'), we can derive other algorithms from it using conjugation, such as the J perm, the F perm, and the Na perm.

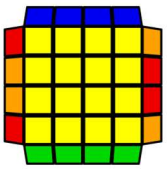
J perm = (R U R' F') : (T). The setup moves the edge sticker C to D while leaving the corner stickers B and C fixed.

F perm = (R' U' F) : (T). This setup moves the edges C to D and B to A. It seems to disrupt the corners, but swapping corner stickers Q and M actually has the same effect as swapping B and C.

Na perm = (R U R' U) : (J). The setup places the D corner and edge to B, and B to C.

Note: The sticker notations are presented in the next 3BLD tutorial.

4×4×4 PPLL Algorithm



The 4×4×4 PPLL algorithm is an even permutation, although it looks like an odd permutation in the 3×3×3 perspective. Assume the yellow face is the front face. Let **(Flip)** = R U R' F R' F' R.

Utilizing the commutator, we can solve this by:

[EU, (Flip)] (Flip) [ED, (Flip)] = EU (Flip) EU' ED (Flip)' ED'

3BLD Methods

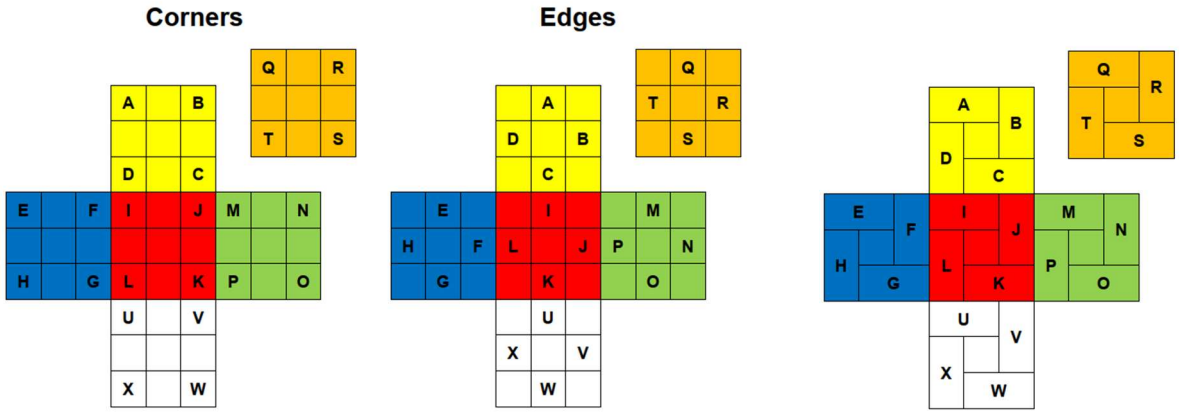
The commutator and conjugate skills are all commonly used in 3BLD advanced methods, such as the E/O (Eka/Orozco) Method and the 3-style method. However, these methods are not covered in this tutorial. The E/O Method is strongly recommended if you want to become an expert in blindfolded solving.

5.6 3BLD Introduction: M2/OP

3×3×3 blindfolded is one of the most fascinating and challenging events in the WCA. It mainly requires memorization skills, but also some advanced techniques as you aim for faster solving times. 3BLD solving is mathematically based on **cycle decomposition**. The solver memorizes cycles of stickers, then executes algorithms that resolve those cycles. This tutorial introduces the basic 2-cycle (or transpositional) methods: **OP Method** (Old Pochmann), and its optimized version, **M2OP**.

In blindfold solving, it is more precise to track **stickers** instead of pieces. An edge has two stickers, and a corner has three stickers. If a piece is in the correct location but flipped or twisted, the piece is correct but the stickers are not. So, we define a set X of sticker positions. A scrambled cube gives a permutation $\pi: X \to X$. The goal is to return every sticker to its home position.

Speffz lettering gives sticker position a letter. These letters are relative coordinates, not fixed colors. 24 letters except Y and Z are used in this system. In blindfolded convention, we usually use yellow top and red front.



It is also important to memorize which stickers constitute a piece:

Corners		Edges	
UFL – DIF	UBL – ARE	UF – CI	BL – RH
UFR – CJM	UBR – BQN	UL – DE	BR – TN
DFL – ULG	DBL – XSH	UR – BM	DF – UK
DFR – VKP	DBR – WTO	UB – AQ	DL – XG
		FL – LF	DR – VO
		FR – JP	DB – WS

Solving Principles

Buffer Cycles

In transpositional solving, we reduce a cycle of pieces into transpositions with a certain sticker. This sticker is called the ***buffer***. When memorizing, we take the sticker coordinates.

Let b be the buffer sticker position. During memorization, look at the sticker currently in the buffer. Suppose this sticker belongs in position x_1. Then the first memo letter is x_1.

After swapping the buffer with x_1, a new sticker enters the buffer. Suppose it belongs to x_2. Continuing gives $x_1, x_2, x_3, ...$ This follows a cycle of the sticker permutation.

Conceptually, buffer solving decomposes the permutation into buffer transpositions:

$$(b\ x_1\ x_2\ \dots\ x_k) = (b\ x_1)(b\ x_2)\dots(b\ x_k)$$

A cycle ends when the buffer piece returns to the buffer. This explains the rule:

Do not include the buffer letter in the memo

If unsolved stickers remain after the buffer cycle closes, the solver starts a new cycle by choosing any unsolved sticker. An easy rule is: When an **old piece** appears (possibly another sticker in the same piece) in the memo, that cycle is closed. The same piece cannot appear more than 2 times.

Extra Swaps

In both OP and M2OP, we first deal with edges, then deal with corners. A pure transposition is impossible on a legal 3×3×3 cube, because one isolated swap changes parity. Therefore, OP and M2OP algorithms are not pure transpositions.

In OP edges, the edge buffer is swapped with a target edge, but a fixed pair of corners is also swapped. If the number of edge targets is even, the fixed corner swap cancels. If the number of edge targets is odd, the fixed corner swap remains. The same idea applies to corners. This gives the parity rule (it works for both OP and M2OP):

$$\#\text{edge letters} \equiv \#\text{corner letters} \pmod 2$$

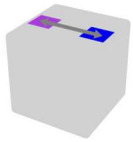
If both are odd, the **parity algorithm** is needed.

In M2OP, each target algorithm also produces a predictable side effect on the M-slice helper positions and centers. These side effects cancel after an even number of edge targets. When the edge and corner memo lengths are odd, a parity algorithm is required before solving the corners.

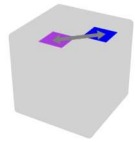
OP and M2OP are considered 2-cycle-style beginner method. Advanced 3BLD methods usually use 3-cycles instead. Since 3-cycles are even permutations, they can often be made cleaner and shorter with commutators.

OP Edges

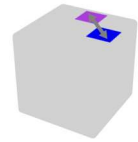
In both OP/M2OP edge and corners, **conjugation** is the key idea to achieve transposition between any two stickers. The setup moves should not disrupt any swapping pieces.



(T Perm)
(R U R' U') (R' F R2 U' R' U') (R U R' F')



(J Perm)
(R U R' F') (R U R' U') (R' F R2 U' R' U')



(L Perm)
U' (R' U L' U2) (R U' R' U2) R L

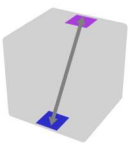
In the OP Method, sticker B is the buffer edge, and the swapping position is A, C, or D. We use 3 algorithms for transposition: T Perm, J Perm, and L Perm. We use L, l, D, d moves to set the target sticker to one of the swapping positions, and perform the corresponding transposition algorithm, then undo the setup moves.

OP Edge Setups: (For reference, not necessary to memorize)

A	**(J)**	**B**	**buffer**	**C**	**(J)**	**D**	**(T)**
E	**L d' L: (T)**	**F**	**D' L: (T)**	**G**	**D l': (J)**	**H**	**d L': (T)**
I	**l': (L)**	**J**	**d2 L: (T)**	**K**	**l': (J)**	**L**	**L': (T)**
M	**buffer**	**N**	**d L: (T)**	**O**	**D' l': (J)**	**P**	**d' L': (T)**
Q	**l: (J)**	**R**	**L: (T)**	**S**	**l: (L)**	**T**	**d2 L': (T)**
U	**l2: (L)**	**V**	**D2 L2: (T)**	**W**	**l2: (J)**	**X**	**L2: (T)**

M2OP Edges

M2OP improves edge solving by using M^2 as the main operation. Since $(M^2)^2 = e$, M^2 has order 2. It acts like a swap operation, but not a pure one.



M2

In M2OP, the buffer is U, the main swapping position is A, and the method also involves the UF and DB edges as an extra swap. These two pieces are considered as special targets. We have the rule:
Whenever you note C, W, I, S as the 2nd, 4th, 6th, etc. letter, change this notation to its opposite:

$$\mathrm{C} \leftrightarrow \mathrm{W}, \mathrm{I} \leftrightarrow \mathrm{S}$$

The mathematical reason is that the extra swap toggles each time M^2 is used. After an odd number of M^2 type edge targets, the helper swap is active. After an even number, it cancels.

M2OP Edge Setups: (You might need to memorize some of them)

A	M2’	B	R U R’ U’: (M2’)	C	U2 M U2 M’	D	L’ U’ L U: (M2’)
E	B L’ B’: (M2’)	F	B L2 B’: (M2’)	G	B L B’: (M2’)	H	L B L’ B’: (M2’)
I	D M’ U R2 U’ M U R2 U’ D’ M2’	J	U R U’: (M2’)	K	buffer	L	U’ L’ U: (M2’)
M	B’ R B: (M2’)	N	R’ B’ R B: (M2’)	O	B’ R’ B: (M2’)	P	B’ R2 B: (M2’)
Q	U B’ R U’ B: (M2’)	R	U’ L U: (M2’)	S	M2’ D U R2 U’ M’ U R2 U’ M D’	T	U R’ U’: (M2’)
U	buffer	V	U R2 U’: (M2’)	W	M U2 M U2	X	U’ L2 U: (M2’)

Parity Inspection

Use a parity algorithm if the number of edge letters and the number of corner letters **are both odd**. The purpose of the parity algorithm is to (1) fix the extra swap resulted from the edge solution, (2) do an extra edge swap in advance between edge sticker A and D, since they are the fix swapping edges during the corner solution.

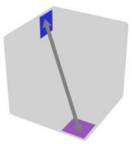
In the OP Method, we use the **Ra Perm: (R U’ R’ U’) (R U R D) (R’ U’ R D’) (R’ U2 R’ U’)**

In the M2OP Method, we use a commutator algorithm with the last move canceled:

D’ L2 D: (M2’) = D’ L2 D M2’ D’ L2 D

Corners

In OP/M2OP, the corner buffer is sticker E, and the swapping position is sticker V. We use R, F, D setup moves to place the target sticker in the swapping position, apply the algorithm, and then undo the setup moves.



(Y) (R U’ R’ U’) (R U R’ F’) (R U R’ U’ R’ F R)

Corner Setups: (For reference, not necessary to memorize)

A	buffer	B	R2’: (Y)	C	F2 D: (Y)	D	F2: (Y)
E	buffer	F	F’ D: (Y)	G	F’: (Y)	H	D’ R: (Y)
I	F R’: (Y)	J	R’: (Y)	K	R’ D’: (Y)	L	F2 R’: (Y)
M	F: (Y)	N	R’ F: (Y)	O	R2’ F: (Y)	P	F D: (Y)
Q	R D’: (Y)	R	buffer	S	D F’: (Y)	T	R: (Y)
U	D: (Y)	V	(Y)	W	D’: (Y)	X	D2: (Y)

Examples

Scramble 1: U2 B D' U' L2 R2 U2 R F R' L U' B' L' B2 F2 L' F2 B2 D' R2 U2 D2 R2 L

We use yellow top and red front in blindfolded convention. Write your notation of solution and apply it (not blindfold) to check if it is correct. We use the OP Method for this one.

Edge Memorization

1st Cycle: B goes to M (old piece and also buffer), therefore C_1 = Nothing
2nd Cycle: C → E → W → U → G → T → P → Q → C (old), therefore C_2 = CE WU GT PQ C
3rd Cycle: L → H → V → F (old), therefore C_3 = LH VF, and edges are completed.

Final edge notation: CE WU GT PQ CL HV F (*Odd*)

Corner Memorization

1st Cycle: E → N → E (old piece and also buffer), therefore C_1 = N
2nd Cycle: D → O → G → X → J → I (old), therefore C_2 = DO GX JI
3rd Cycle: V → P (old, actually it twists clockwise), therefore C_3 = VP, and edges are completed.

Final corner notation: ND OG XJ IV P (*Odd*)

Final Notation = (CEW UGT PQ CL HVF) (Parity) (N DOG X JIV P)

Scramble 2: x2 y D' F2 L2 U' R2 U B2 D U F' R F2 L B F U2 L U2 F R'

Use this one to practice M2OP edge solution.

There is only one cycle: *C* = HI EF BO SA GN P (*Odd*) (Change I to S)

The final solution is: **(H SEF BO SAG NP) (Parity) (Corner solution)**

Useful skills: The final notation itself seems long, even without applying it. You can regroup them to make them easier to memorize. In the example above, you can group the letters "cew", "ugt", "dog", and "jiv". Getting familiar with the coordinates and lettering scheme is also important for beginners.

5.7 3FMC Introduction: NISS

FMC is the event where solvers try to find the shortest solution, not the fastest execution. Because of that, the mathematics of inverse states and subgroups becomes especially useful. This tutorial introduces **NISS**, or Normal-Inverse-Scramble-Switch, as a way to use both the scramble and its inverse during a solve.

In an official WCA FMC attempt, competitors receive a printed scramble sequence and paper. They have 60 minutes to find and write a solution. Competitors may use physical cubes to analyze the scramble, test partial solutions, and compare normal and inverse states.

For convenience, we use the following notations and definitions:

1. scr means scramble, and the scramble algorithm is denoted as N.
2. $A = \mathrm{sol}(N)$ if A is a solution algorithm for scramble N, i.e. $NA = e$.
3. $\mathrm{psol}(N)$ is a partial solution algorithm for scramble N.
4. $A' = A^{-1}$ is the inverse algorithm of A. The inverse of the scramble algorithm is denoted as I, which means $N^{-1} = I$.

Let N be a scramble, consider the scramble N^{-1}. Let $S = \mathrm{sol}(N^{-1})$, which means $N^{-1}S = e$. Then $NS^{-1} = e$, which means $S^{-1} = \mathrm{sol}(N)$. This is the core of NISS. It uses the fact that solving the inverse scramble can also give information about solving the original scramble. You can "switch" to an inverse working state once you find the current one hard to deal with.

An important fact is that after switching, **solved pieces stay solved**. Suppose a piece or block P is solved in a state g. This means $g(P) = P$. Then in the inverse state, $g^{-1}(P) = P$ also holds. Therefore, if you solve a cross, block, pair, or EO structure on one side, switching does not destroy it. It only changes the viewpoint.

Solving Process

Suppose N is the scramble we are dealing with, and I is the inverse of the scramble.

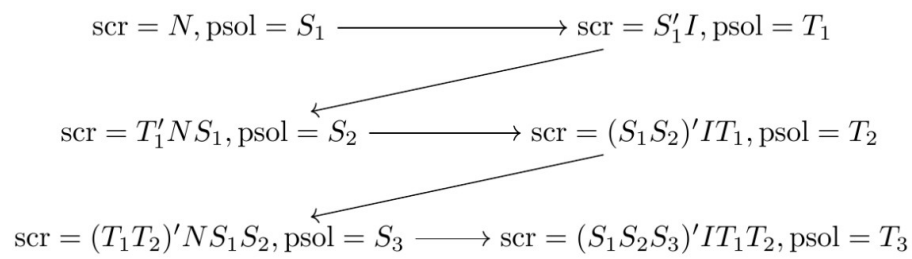
After the first normal-side work $S_1 = \mathrm{psol}(N)$, the inverse state is $(NS)' = S_1'I$. Therefore, to switch to the inverse working state, use scramble $\mathrm{scr} = S_1'I$.

After the first inverse-side work $T_1 = \mathrm{psol}(S_1'N')$, switch back to the normal working state, $\mathrm{scr} = (S_1'N'T_1) = T_1{}'NS_1$. Repeat this process until the cube is solved.

If the cube is solved after $S_n = \mathrm{sol}[(T_1 \dots T_{n-1})'N(S_1 \dots S_{n-1})]$, then $(T_1 \dots T_{n-1})'N(S_1 \dots S_{n-1})S_n = e$. Simplify this expression, $N = (T_1 \dots T_n)(S_1 \dots S_{n-1})'$, which means $\mathrm{sol}(N) = (S_1 \dots S_n)(T_1 \dots T_{n-1})'$.

If the cube is solved after $T_n = \mathrm{sol}[(S_1 \dots S_n)'N(T_1 \dots T_{n-1})]$, $\mathrm{sol}(N) = (S_1 \dots S_n)(T_1 \dots T_n)'$, by a similar argument.

This graph gives a clean solving process:


$\text{scr} = N, \text{psol} = S_1$
$\text{scr} = S_1' I, \text{psol} = T_1$
$\text{scr} = T_1' N S_1, \text{psol} = S_2$
$\text{scr} = (S_1 S_2)' I T_1, \text{psol} = T_2$
$\text{scr} = (T_1 T_2)' N S_1 S_2, \text{psol} = S_3$
$\text{scr} = (S_1 S_2 S_3)' I T_1 T_2, \text{psol} = T_3$

In a real solve, the process is usually:

1. Find the inverse of the scramble algorithm, I.
2. For each scramble, solve something as long as your solution is short enough. Carry out all your normal solving skills.
3. If you are not confident with the current continuation, use another cube to make the inverse working state. Compare them and decide where to continue.
4. If your final partial solution is S_n, then $\text{sol}(N) = (S_1 \dots S_n)(T_1 \dots T_{n-1})'$.
5. If your final partial solution is T_n, then $\text{sol}(N) = (S_1 \dots S_n)(T_1 \dots T_n)'$.

Some other tips:

1. Switching does not affect solved pieces, but would usually scramble up all unsolved pieces. So, make sure all targeted pieces are completely solved before switching.
2. It is generally very beneficial to consider the Petrus Method, especially the 2×2×2 and 2×2×3 part. You can also borrow some ideas of ZZ or Domino Reduction if you see a good opportunity. Memorizing ZBLL algorithms also helps in FMC solving.
3. As you progress to the end of your solution, the scramble algorithm would become more and more complicated. Instead, you can solve the scramble of the last working state and use this solution as the next scramble.
4. Look for potential move cancellation during solving.

Example

$N =$ **R' U' F L' B2 L' U2 L B2 L' U2 L2 D2 B L U L2 D2 U L' D2 L2 R' U' F**

$I =$ **F' U R L2 D2 L U' D2 L2 U' L' B' D2 L2 U2 L B2 L' U2 L B2 L F' U R**

Try it on your own first! There are two example solutions on the next page. See if you can do better than those.

In a competition, you need to write out the inverse algorithm by yourself.

Example Solution 1 (Simple CFOP)

scr#1 $= N$, there is no easy cross, so switch to I directly, $S_1 =$ **None**.

scr#2 $= I$, $T_1 =$ **F2 L2 F' L' F'** (5) // Orange Cross, but F2L #1 = 7 steps, switch.

scr#3 $= T_1'NS_1$, $S_2 =$ **U' R2 U2 R U' R' F D2 F' D2** (10) // First 2 pairs, switch.

scr#4 $= (S_1S_2)'IT_1$, $T_2 =$ **R' D R D'** (4) //3rd pair. F2L #3 = 8 steps, switch.

scr#5 $= (T_1T_2)'NS_1S_2$, $S_3 =$ **R F R2 F2 U F U'** (7) // Last pair. OLL is long, switch.

scr#6 $=$ sol(scr#5 S_3), $T_3 =$ **F' U' R' U R U' R' U R F** (10) // OLL, switch.

scr#7 $=$ sol(scr#6 T_3), $S_4 =$ **U2 L U L' U B2 D' R D B2** (10) // PLL, the cube is solved.

sol(N) $= (S_1S_2S_3S_4)(T_1T_2T_3)'$ = (U' R2 U2 R U' R' F D2 F' D2 R F R2 F2 U F U' U2 L U L' U B2 D' R D B2) (F2 L2 F' L' F' R' D R D' F' U' R' U R U' R' U R F)' (*move cancellation*)

Final Solution: U' R2 U2 R U' R' F D2 F' D2 R F R2 F2 U F U L U L' U B2 D' R D B2 F' R' U' R U R' U' R U F D R' D' R F L F L2 F2 (45 HTM)

Example Solution 2 (Petrus)

scr#1 $= N$, switch directly, $S_1 =$ **None**.

scr#2 $= I$, $T_1 =$ **B U' F2 L** (4) // 2×2×2 Block. Switch.

scr#3 $= T_1'NS_1$, **F U R F** // 2×2×3 Block. Compare with inverse working state.

Continue, use PSF2L along with Petrus EO: **U R' D B U B' D2 U' F' U' F D** // F2L + EO.

$S_2 =$ **F U R F U R' D B U B' D2 U' F' U' F D** (16). Switch.

scr#4 $=$ sol(scr#3 S_2), $T_2 =$ **B U B' U F' U2 B U B' U2 F B U2 B' U2** (15) // ZBLL. Solved.

sol(N) $= (S_1S_2)(T_1T_2)'$

Final Solution: F U R F U R' D B U B' D2 U' F' U' F D U2 B U2 B' F' U2 B U' B' U2 F U' B U' B' L' F2 U B' (35 HTM)

Chapter Summary

This is a summary of important concepts in this chapter.

1. A *group* is a set together with an operation satisfying four axioms: closure, associativity, existence of identity element, and existence of inverse.
2. The set of legal Rubik's Cube state forms a group: $\mathcal{R} = \langle U, D, L, R, F, B \rangle$.
3. A *permutation* is a rearrangement of a set. A *cycle* shifts elements in a circular manner. A *transposition* is a swap between two elements.
4. A permutation is considered *even* if it can be expressed as a product of an even number of transpositions; *odd* if it can be expressed as a product of an odd number of transpositions. The parity of a k-cycle is $\operatorname{sgn}(a_1 \dots a_k) = (-1)^{k-1}$.
5. A cube state can be described by four pieces of data: $(\sigma_c, \omega_c, \sigma_e, \omega_e)$, where σ_c = corner permutation, ω_c = corner orientation, σ_e = edge permutation, ω_e = edge orientation.
6. Legal states obey three invariants:
$$\text{(i)} \sum_{i=1}^{12} e_i \equiv 0 \pmod 2; \ \text{(ii)} \sum_{i=1}^{8} c_i \equiv 0 \pmod 3; \ \text{(iii)} \operatorname{sgn}(\sigma_e) = \operatorname{sgn}(\sigma_c).$$
7. The *size* of the group is the number of elements, or the cardinality, of the group. For the Rubik's Cube group $\mathcal{R}$,
$$|\mathcal{R}| = \frac{8! \times 3^8 \times 12! \times 2^{12}}{3 \times 2 \times 2} = 43{,}252{,}003{,}274{,}489{,}856{,}000$$
8. A *subgroup* is a group contained within another group under the same operation. A *generator set* is a set of moves that produces a group: $G_0 = \langle U, D, L, R, F, B \rangle$. Subgroups explain methods such as Domino Reduction: $G_1 = \langle U, D, L^2, R^2, F^2, B^2 \rangle$.
9. The *order* of an element g is the smallest positive integer n such that $g^n = e$. The order of any element in $\mathcal{R}$ is finite.
10. The *coset* of subgroup is the same group moved to another location, either identical or disjoint, in the full group.
11. **Lagrange's Theorem:** If G is finite and H is a subgroup of G, then $|G| = m|H|$ for some positive integer m. The cosets of any subgroup partition the group into disjoint sets, and that each of these partitions contains the same number of elements.

12. A *Cayley graph* is a visual way to represent a group and its generators. The *distance*, $d = (g, h)$, is the minimum number of moves needed to get from state g to state h. The diameter, $D = \max_{g \in \mathcal{R}} d(e, g)$, is the maximum distance between the identity state and any other state.
13. **God's Algorithm** is an ideal algorithm that always solves a cube in the fewest possible moves. **God's Number** is the worst-case length of such an optimal solution:
$$\text{God's Number} = \max_{g \in \mathcal{R}} d(e, g) = 20 \text{ HTM}$$
Most random cube states are around 17 to 18 moves from solved.
14. A **Demigod's Number** is a weaker bound that is easier to understand and reproduce.
15. A *commutator* measures how two elements fail to commute. $[A, B] = ABA^{-1}B^{-1}$.
16. A *conjugate* is $A : B = ABA^{-1}$. A is the setup, B is the main algorithm, and A^{-1} undoes the setup. A *conjugated commutator* is $A : [B, C] = ABCB^{-1}C^{-1}A^{-1}$.
17. A *affects* element x if $Ax \neq x$. The *support* of A, written $\text{supp}(A)$, is the set of all elements affected by A: $\text{supp}(A) = \{x : Ax \neq x\}$. If $x \notin \text{supp}(A)$, then $Ax = x$.
18. **3-cycle Commutator Theorem:** Let $[A, B] = ABA^{-1}B^{-1}$ be a commutator. Suppose $\text{supp}(A) \cap \text{supp}(B) = \{x\}$, and $Ay = x$, $Bz = x$, where x, y, z are distinct elements. Then, $[A, B] = (y\ x\ z)$. That is, if A and B affect exactly one sticker x, A moves y to x, B moves z to x, then the commutator cycles exactly three stickers: $y \to x$, $x \to z$, $z \to y$.
19. Two common forms of a practical commutator are [interchange, insert] and [interchange, insert']. A conjugated commutator is needed if there is no one-move interchange.
20. 3BLD solving is mathematically based on **cycle decomposition**. **OP** and **M2OP** are transpositional solving methods that reduce a cycle into transpositions.
21. **NISS**, or Normal-Inverse-Scramble-Switch, is a fundamental method for FMC solving that uses the fact that solving the inverse scramble can give information about solving the original scramble. After switching to the inverse working state, solved pieces stay solved.

Source-Use Notes

Most explanatory prose in this book was written by the author. External sources were used primarily for algorithms, case organization, diagrams, historical information, competition data, and mathematical results.

- **CubeRoot and tutorials by Ruimin Yan:** notation and selected diagrams on pp. 3 and 7; two-by-two methods on pp. 10–12; F2L, OLL, PLL, CFCE, Roux, advanced methods, and algorithm collections on pp. 15–78; big-cube and twisty-puzzle material on pp. 80–96 and 100–102; commutators on pp. 114–117; blindfolded solving on pp. 118–122; and NISS on pp. 123–125.
- **CubeSkills:** selected beginner last-layer and four-look last-layer diagrams on pp. 6 and 8–9.
- **Ruimin Yan's short-form solve analyses:** world-record reconstructions and commentary on pp. 45–46.
- **Sengso Cube:** FTO diagrams and algorithms on pp. 97–99.
- **MIT, Zuo and Lai, and Ruimin Yan:** group theory, permutations, commutators, and cube mathematics on pp. 104–117.
- **Kociemba and Rokicki et al.:** Domino Reduction, God's Number, and move-distance data on pp. 108–113.
- **Merino and Subercaseaux:** Demigod's Number on p. 113.
- **World Cube Association:** competition information in the Preface, record tables, official event information, and competition procedures.
- **ZZ Method, Speedsolving Wiki, and tutorials by Ruimin Yan:** ZZ and Petrus material on pp. 72–78.

References

1. Akkam, Alia. “Rubik’s Cube Inventor on 50 Years of the Cult Puzzle.” *Wallpaper**, 20 May 2024, https://www.wallpaper.com/design-interiors/rubiks-cube-anniversary-interview-erno-rubik. Accessed 10 July 2026.
2. FewestMoves. “How to Do Domino Reductions: The Secret They Don’t Want You to Know.” *YouTube*, https://www.youtube.com/watch?v=oAbPQOfui2U. Accessed 10 July 2026.
3. J Perm. “Rubik’s Cube: How to Learn the CFOP Speedcubing Method.” *YouTube*, https://www.youtube.com/watch?v=MS5jByTX_pk. Accessed 10 July 2026.
4. Jha, Aditya. “Cube Theory; The Fascinating Math Behind the Rubik’s Cube.” *Cubing Content*, 30 May 2023, https://cubingcontent.com/2023/05/30/cube-theory-the-fascinating-math-behind-the-rubiks-cube/. Accessed 10 July 2026.
5. Kociemba, Herbert. “Two-Phase Algorithm and God’s Algorithm: God’s Number Is 20.” *Kociemba.org*, 2018, https://kociemba.org/moves20.htm. Accessed 10 July 2026.
6. Merino, Arturo, and Bernardo Subercaseaux. “A Demigod’s Number for the Rubik’s Cube.” *arXiv*, 30 Dec. 2024, arXiv:2501.00144, https://arxiv.org/abs/2501.00144.
7. “The Mathematics of the Rubik’s Cube: Introduction to Group Theory and Permutation Puzzles.” *Massachusetts Institute of Technology*, 17 Mar. 2009, https://web.mit.edu/sp.268/www/rubik.pdf.
8. Rokicki, Tomas, Herbert Kociemba, Morley Davidson, and John Dethridge. “The Diameter of the Rubik’s Cube Group Is Twenty.” *SIAM Journal on Discrete Mathematics*, vol. 27, no. 2, 2013, https://doi.org/10.1137/120867366.
9. SengSo Cube. “[FTO Tutorial and Algorithm Diagrams].” *WeChat Official Account*, n.d.
10. “SpeedSolving the Rubik’s Cube.” *Speedsolving Wiki*, https://www.speedsolving.com/wiki/index.php?title=Main_Page. Accessed 10 July 2026.
11. World Cube Association. “Changes to the WCA’s List of Official Events (June 2026).” *World Cube Association*, June 2026, https://www.worldcubeassociation.org/posts/changes-to-the-wca-s-list-of-official-events-june-2026. Accessed 10 July 2026.
12. World Cube Association. “Records.” *World Cube Association*, https://www.worldcubeassociation.org/results/records. Accessed 10 July 2026.
13. World Cube Association. “WCA Regulations.” Version 1 Apr. 2026, https://www.worldcubeassociation.org/regulations/. Accessed 10 July 2026.

14. Yan, Ruimin. *CubeRoot*. https://cuberoot.me. Accessed 10 July 2026.
15. Yan, Ruimin. “[Selected Tutorials on Speedcubing Methods, Commutators, Blindfolded Solving, and NISS].” *WeChat Official Account*, n.d.
16. Yan, Ruimin. “[Short-Form Analyses and Reconstructions of Rubik’s Cube World-Record Solves].” Online video series, n.d.
17. Zemdegs, Feliks. *CubeSkills*. https://www.cubeskills.com/. Accessed 10 July 2026.
18. Zuo, Bowen, and Hongyu Lai. “From Chaos to Order: A Group Theory Approach to the Rubik’s Cube.” *Proceedings of the 7th International Conference on Algorithms, Computing and Systems*, ACM, 2024, https://doi.org/10.1145/3631908.3631933.
19. “ZZ Method.” *Google Sites*, https://sites.google.com/view/zzmethod/home. Accessed 10 July 2026.

Image and Diagram Credits

Page numbers refer to the printed page numbers in this book. Unless otherwise indicated, the page layout, captions, annotations, tables, arrows, and explanatory graphics were created or arranged by the author.

Cover and title page. AI-generated artwork produced under the author's direction, 2026.

Table of Contents and chapter-opening artwork, pp. 1, 14, 44, and 79. AI-generated artwork produced under the author's direction, 2026.

Page i — Photograph of Ernő Rubik. Photograph credited as "Courtesy Ernő Rubik" in Alia Akkam, "Rubik's Cube Inventor on 50 Years of the Cult Puzzle," *Wallpaper**, 20 May 2024.

Page i — Twisty-puzzle collage. AI generated work produced under author's direction, 2026.

Page ii — World Cube Association logo. Courtesy of the World Cube Association; used in accordance with the WCA Logo Usage Policy.

Page ii — CubeRoot logo. Source: CubeRoot, created by Ruimin Yan.

Page 2 — Structure and color diagrams. Created by the author.

Pages 3 and 7. Cube notation, cross, F2L, and keyhole diagrams adapted from CubeRoot.

Pages 4–6. Beginner-method diagrams created by the author, except for selected last-layer case graphics on page 6, which were adapted from CubeSkills.

Pages 8–9. Four-look last-layer case diagrams adapted from CubeSkills.

Pages 10–33. Two-by-two, F2L, OLL, PLL, and one-handed algorithm diagrams adapted from CubeRoot.

Page 34 — CFOP promotional graphic. Adapted from J Perm's YouTube video "Rubik's Cube: How to Learn the CFOP Speedcubing Method."

Pages 34–78. CFCE, Roux, advanced F2L, OLS, EOLS, VH, ZB, ZZ, Petrus, and associated algorithm diagrams adapted from CubeRoot and tutorials by Ruimin Yan.

Pages 80–96 and 100–102. Big-cube, Pyraminx, Skewb, Square-1, and Megaminx diagrams adapted from CubeRoot.

Pages 97–99 — FTO diagrams. Adapted from an FTO tutorial published by SengSo Cube through its WeChat Official Account.

Page 103 — Cube Theory graphic. Adapted from Aditya Jha, “Cube Theory; The Fascinating Math Behind the Rubik’s Cube,” *Cubing Content*, 30 May 2023.

Page 110 — Domino Reduction illustration. Adapted from the FewestMoves YouTube video “How to Do Domino Reductions: The Secret They Don’t Want You to Know.”

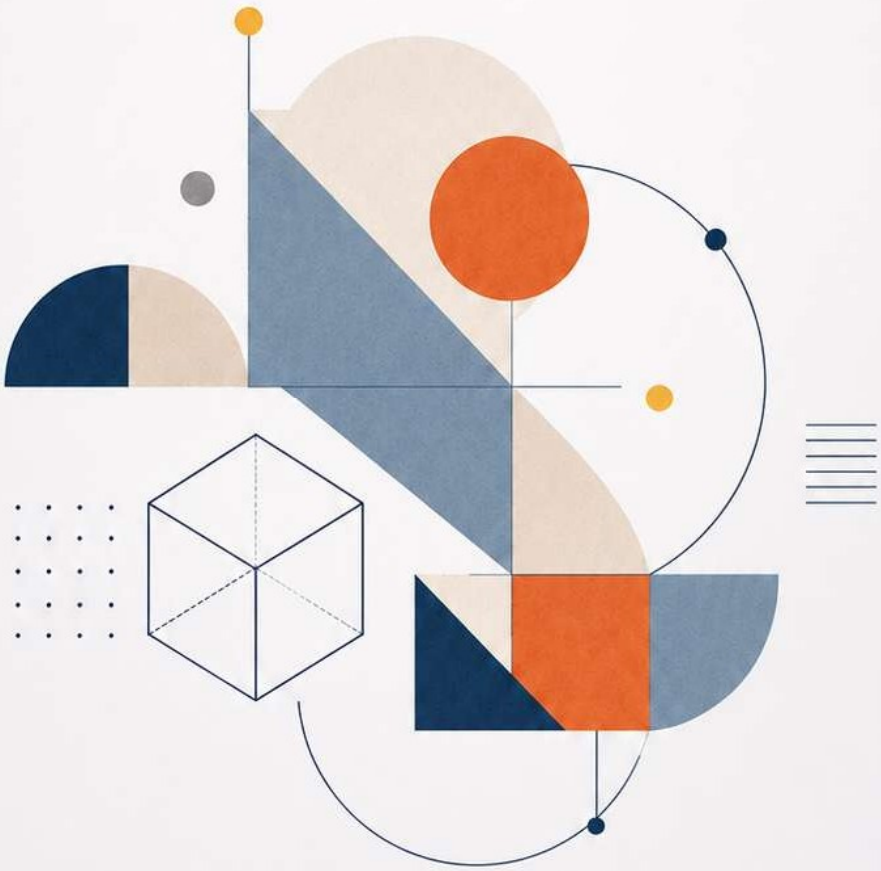
Page 111 — Cayley graph. Adapted from *The Mathematics of the Rubik’s Cube: Introduction to Group Theory and Permutation Puzzles*, Massachusetts Institute of Technology, 17 March 2009.

Pages 112–113 — God’s Number and Demigod’s Number tables. Created by the author using information from Herbert Kociemba, CubeRoot, Rokicki et al., and Merino and Subercaseaux.

Pages 114–116. Commutator illustrations and examples created by the author.

Page 117 — Four-by-four PPLL diagram. Adapted from a CubeRoot commutator tutorial by Ruimin Yan.

Pages 118–122 — Blindfolded-solving diagrams. Adapted from CubeRoot and blindfolded-solving tutorials by Ruimin Yan.

Pages 123–125 — NISS diagrams. Adapted from an NISS tutorial by Ruimin Yan.

Rubik’s Cube, World Cube Association, CubeRoot, CubeSkills, QiYi, and other product or organization names and logos are the property of their respective owners. Their appearance in this educational work does not imply endorsement.